Semantic models for texturing volume objects

Shen, Peiyi

Award date:
2007

Awarding institution:
University of Bath

Semantic Models for Texturing Volume Objects

submitted by

Peiyi Shen

for the degree of Doctor of Philosophy

of the

University of Bath

July 2007

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

SEMANTIC MODELS FOR TEXTURING VOLUME OBJECTS

Submitted by Peiyi Shen
for the degree of Doctor of Philosophy
of the University of Bath
July 2007

To My Wife , My Parents , and My Younger Sister.

Acknowledgements

First, my sincere thanks my supervisor, Professor Philip Willis. It is his encouragement, patience and fatherly love to students that lead me go through this three years hardworking time. I am very lucky to be one of his students. I am grateful for all of his keen insights on the research fields and detailed advice that shape the direction of this work.

I want to acknowledge all the colleagues and friends in the Media Technology Research Centre at the University of Bath. My special credit goes to Mr. Yi-Ze Song, Dr. John Collomosse, Dr. Peter Hall, Dr. Adam Batenin, Dr. William Naylor, Dr. Emmanuel Tanguy, Mr. Adam Dziedzic, Dr. Sirapat Boonkrong, Mr. Tom Crick, Mr. Martin Brain, Mr. Jonty Needham, Mr. Mark Wood, Dr. Matthew Collinson, Dr. Emma Jones, Dr. Owen Cliffe, Dr. Mark Cahill, Mr. Jim Grimmett, Mr. Martin Post and Mr. Chuan Li for their kind support and help.

I would like to thank my parents and my younger sister for their immeasurable support and encouragement. In particular, I would like to thank my wife for her unconditional love.

Finally, I would like to thank our EPSRC project collaborators at the University of Wales, Swansea, for their valuable support and advice: Professor Min Chen, Dr. Mark Jones, Mr. Shoukat Islam and Mr. Simon Walton; and especially thanks to Dr. Andrew Winter for his support with VLIB. I am especially grateful to Prof. Min Chen and Dr. Mark Jones. Their exceptional insights of science, their talents with understanding of research, always encourage me and lead me go through the difficulties.

The lovely sunshine and the beautiful scene in Singapore are unforgettable. However, the dream in my life is still beating my heart, waiting for me to make it become reality . Surely, I do not regret a thing.

Declaration

The research presented in this thesis was conducted by the author.

All views expressed in this thesis are those of the author, and do not reflect those of the University of Bath.

Summary

Texturing volume datasets is an important topic in scientific visualisation, surgical planning, clinic application and medical education. A good texture model should not only present the abstract information to the target users but also give realistic appearances to volume datasets. It should be developed on the basis of domain knowledge, user requirement and human factors. Unfortunately, segmented, classified and clustered volume objects are conventionally annotated with non-realistic colours to make them different to their neighbours.

In addition, highly detailed textures are subjugated to making crucial features immediately noticeable. The subsequent loss of realism causes conventional visualisation and illustration systems to become unconvincing and unacceptable.

Motivated by the demand of *realism* for volume objects in medical and entertainment applications, we focus on image based realistic volume rendering techniques.

Photos / images provide a variety of realistic visual effects. So we would like to use these resources to texture map volume objects. Textured volume objects could thus present sensible appearances to end users.

First, we propose a *projective model* for texturing volume objects. We use *semantic constraints* to guide the texture projection within volume objects. We develop *splitting layers* to control texture penetration and volume self-occlusion.

Second, texture projection is guided through a *Multi-Dimensional-Scaling (MDS)* based *Linear-Weighted Laplacian Smoothing (LWLS)* method , which flattens plenoptic and cel based intermediate templates for texture mapping. In particular, the smoothed intermediate template can be used to reduce sheared textures, by providing multi-resolution representations of texcels. The smoothed intermediate template preserves the areas of surfaces on volume objects. Highly detailed close-ups are offered based on multi-resolution representations.

Third, the novel *projective model* bridges image based realistic appearances and volume based datasets. It lifts the restrictions (non-realistic appearances, pseudo-textures, etc.) of conventional Non-Photo-Realistic (NPR) based volume annotation.

In summary, the presented models have the potential to lead to several important applications which include: *arbitrary resolution enhancement*, *multi-layer isosurface annotation*, *direct splitting*, and *transfer functions*. More importantly, the *realism* in photos / images could thus be transferred onto volume datasets, to meet the target of perception and cognition based visualisation.

We do not build any explicit mesh models for texture manipulation and volume manipulation. The algorithms are implemented in continuous space. The texture projection pipeline is implemented based on volume data sets. Therefore, our system could also be used as a benchmark for testing volume visualisation systems.

The work has been presented at "Vision, Video, and Graphics 2005, (Edinburgh)", "The 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2005, (Dunedin)" and the 26th Eurographics Conference (Dublin, Ireland, 2005).

Contents

List of Figures

List of Tables

Chapter 1

Introduction: DSOR Representations and Challenges in Volume Visualisation

1.1 DSOR Representation and Realistic Appearances

Visualisation research focuses on exploring Discretely Sampled Object Representations (DSOR) to provide stunning images and effective interactive systems [1]. The variety of scanned volume data sets, for instance, of human bodies, animals, art works, vegetables, etc., are now commonly used in computer animation, surgical planning and medical education.

"Do expert reviews work?" [2],"Is beauty enough?", "Do the fascinating demos of visualisation demonstrate their usefulness rather than just visual fraud and illusion?", these doubts draw our attention to the obvious gap between research-focus visualisation systems and realism-focus medical or entertainment applications.

Most existing scientific visualisation techniques focus on making the data intelligible to target users. As a result, pseudo-colour techniques are combined with selector mechanisms to abstract interesting features of the data. However, small but important features of the physical objects might be lost. Visualisation might be informative, but it is unacceptable to novices, due to the loss of natured looking appearance of physical objects.

In contrast the research presented in this thesis is driven by recreating the appear-

ance as realistically as possible, thus extending volume datasets into many areas currently served by traditional techniques, such as surgical training and medical education.

We recognise the importance of giving a volume data set natural appearance. Therefore, this thesis addresses a problem of texture mapping a volume data set: to find an appropriate representation of digital images that provides a link between 2D, 2.5D, 3D texture models and the 3D volumetric data set. We believe it can be the foundation of many other image-related applications in the areas of volume visualisation.

The intuition behind our image based representation model is the importance of the roles played by images. Therefore we believe the contribution of this thesis can provide an appropriate tool to bridge the gap between hidden structures of volume datasets and the 2D, 2.5D and 3D texture information. Applications that need realism based volume objects, such as medical training, surgical planning, can be drawn from this model.

The work described here is funded by the UK EPSRC research grant "Volume Animation". EPSRC also support a related grant at the University of Swansea. In order to develop a coherent modelling scheme that brings together discretely sampled object representations, to develop algorithms and methods for different stages of animation pipeline, and to develop a volume-based computer animation toolkit, our collaborators focus on developing novel spatial constraints and models for splitting and animating volume objects. Here, my research focuses on texture mapping volume objects and providing colour fields (scalar fields) which can be used in the animation pipeline. The webpage of the project can be found at: http://www.cs.bath.ac.uk/van.

1.2 Contributions

This thesis gives a possible solution to realistic rendering of volume datasets. Here, "realistic" refers to transferring realism in images or photos onto volume objects, especially textures with detailed information in real appearances of physical objects [3, 4].

First, we present an imaged based approach to texture mapping volume datasets [39, 40]. The method is based on a projective pseudo-solid texture model. A rendered intermediate template for texture warping is needed.

The rendered intermediate template is based on projective mapping. Therefore, texcels will be smeared over a relatively large area if they are projected onto a sur-

face which is almost parallel to the direction of projection. Our second contribution is the solution to this problem by flattening the intermediate template using Multi-Dimensional-Scaling (MDS) and Linear-Weighted-Laplacian-Smoothing (LWLS).

We focus on realistic texture mapping and high quality close-ups for texturing volume objects. In addition, we introduce a meshless model for texturing volume objects, that is, directly manipulating point clouds rather than construct a mesh model as an intermediate step for texture mapping and annotation.

In summary, the main contributions of this thesis are:

- A projective texturing system which links 2D photos / images and 3D volume datasets. The realistic texture featuring in photos / images can therefore be transferred onto volume datasets.
- A point cloud smoothing method using MDS-based LWLS, which allows us to flatten the intermediate template and thus match textures to the shape of the object. We can prevent texture twisting or tangling.
- A method to reduce shear texture, by using the flattened texture intermediate template.
- A multi-resolution representation of the flattened intermediate template, rendered by directly manipulating point clouds rather then warping an abstracted mesh model as an intermediate step.
- An algorithm to texture high quality close-ups, using tiled flattened patches.
- A volume splitting model (based on projective semantic layers and a texture atlas) to effectively control texture penetration and volume self-occlusion.

1.3 Organisation

The rest of this thesis is organised as follows: A general review of texturing volume objects and new challenges for visualisation tasks are given in Chapter 2. Scalar field based projective texture models are introduced in Chapter 3. The problem of the texture shear effect that leads us to develop the surface flattening techniques (Multi-Dimensional-Scaling) is described in Chapter 4. The problem of texture tangling

which leads us to develop the sufficient-condition solution (Linear-Weighted-Laplacian-Smoothing) is described in Chapter 5. Since we directly flatten the point cloud within volume objects rather then warp a constructed mesh model for texture indexing, multi-resolution representations of flattened intermediate templates are also given in Chapter 5. The novel split models which are developed to control texture penetration and volume self-occlusion are offered in Chapter 6. Finally, we make concluding remarks in Chapter 7. The detailed outline is given below.

- **Chapter 2: Challenges and Background: Texturing Volume objects** This chapter will briefly review the history and the scientific background of the techniques of texturing and annotating volume objects. In addition, perception and cognition based visualisation techniques are also discussed here. It is worth pointing out that perception based visualisation has become an important research topic in the recent years. Our initial target is to transfer realism in photos/images onto volume objects, to preserve the perceptual texture information which is subjugated in conventional annotations.

- **Chapter 3: Projective Texture Models** This chapter starts by describing the new projective texture models for texturing volume objects. It also presents the generic concept of semantic constraints which are used for splitting volume objects, spatially or logically; whereas, traditional field functions or spatial transfer functions focus on solving specific visualisation problems but providing generalised algorithms. Multi-level volume rendering, multi-texture models, varieties of field functions and spatial transfer functions could thus be consolidated into a flexible and practical rendering system under such a semantic framework. The projective texture primitives and illustrative colour transfer techniques (from illustrative photos / image to volume data) are also described here.

- **Chapter 4: Multi-Dimensional-Scaling Models (MDS)** This chapter shows how the Multi-Dimensional-Scaling (MDS) method can be applied to projective texture models to flatten point clouds within volume objects. It shows how the texturing results can be improved by flattening the texture intermediate template. Traditional MDS smoothing method, the geodesic distance MDS method and the graph model based Euclidean distance MDS method are discussed here.

- **Chapter 5: Linear-Weighted-Laplacian-Smoothing (LWLS) for Flattening Point Clouds** This chapter offers a Linear-Weighted-Laplacian-Smoothing (LWLS) model to flatten point clouds within volume objects. Compared with MDS smoothing methods described in the previous chapter, we demonstrate that the proposed MDS-based LWLS method prevents the tangling of flattened point clouds. Here the MDS method is used to calculate smoothing weights and boundary conditions in LWLS. Edge length based smoothing weights are discussed.

  This chapter also introduces the multi-resolution representations of flattened intermediate templates. Using MDS-based LWLS smoothing control, we render the intermediate template using volume rendering techniques, i.e., direct volume rendering (DVR) and direct surface rendering (DSR). This time, the resolution of sampling positions are directly controlled by the size of the areas of flattened intermediate surfaces. The larger the area of the flattened surface, the more sampling positions are allocated. Smoothing control and texture annotation are based on direct manipulation of point clouds. We do not build any explicit mesh models for data representation, neither for volume data sets nor for texture models.

- **Chapter 6: Projective Masking Fields** Novel splitting models are offered in this chapter, to solve the problem of volume self-occlusion and texture penetration. The space is split into different semantic layers and so each of them can be textured independently. A texture atlas can be used to annotate each layer accordingly.

  This chapter also gives a review of consolidating different modules into a system. Some more results of the proposed visualisation and annotation techniques are given. A possible application of intermediate templates is also investigated.

- **Chapter 7: Conclusions and Future Work** We finally make our concluding remarks and provide an overview of future research orientation.

1.4 Terminology

We will frequently use the word "realistic" to refer to the real appearances of physical objects. This not only refers to the colour realism in texture images, but also to the texture features which contribute to the human perceptual and cognitive process. Unfortunately, such texture features are often hidden by conventional visualisation techniques. In addition, in terms of cognitive science, "Intuitive" refers to "immediate apprehension or cognition" [5], i.e, the direct understanding of the communicated information.

We also use the word "isosurface" to refer to the point clouds or level sets within volume objects. We do not assume the mesh model is the eponym of these discrete point sets, whereas the graphics community usually does.

1.5 Datasets and Software

The volume graphics API (*VLIB*: http://vg.swan.ac.uk/vlib/) is used to construct the volume rendering pipeline. The original volume graphic system and API functionalities were described in Dr. Andrew S. Winter's PhD thesis [6]. The development of the semantic-projective texture model, semantic volume splitting, multi-dimensional scaling and Laplacian smoothing, multi-resolution representations of intermediate template, and data-dependant interpolation were developed as additional API functions by the author of this thesis.

The matrix operations in MDS based LWLS smoothing, the colour transferring between illustrative photos/images and slices of volume data were implemented using MATLAB(7.01).

The volume fish data was downloaded from the volume library constructed by Dr. Stefan Roettger at the Computer Graphics Lab of the University of Erlangen (http://www9.cs.fau.de/Persons/Roettger/library/). The CTHead data was downloaded from VLIB datasets. The other datasets not explicitly mentioned are constructed by the author of this thesis.

Chapter 2

Challenges and Background: Texturing Volume Objects

This chapter will briefly review the history and the scientific background of the techniques of texturing volume objects. In addition, perception and cognition based visualisation techniques are discussed here. Perceptual visualisation has become an important research topic in recent years. Our initial targets coincide with this, by preserving realism, i.e., perceptual information in volume visualisation. Realism is preserved by directly transferring photographs or images onto volume objects. As we will explain in the following sections, we focus on developing image based realistic volume rendering techniques, rather then constructing neuropsychology and neurology-based perceptual models for volumetric objects.

2.1 DSOR: Challenges in Texturing Volume Objects

It is commonly recognised that digital imaging technology is rapidly become an effective way of collecting data and information. Acknowledged by the enormous number of captured images, screened videos, scanned volume datasets and point datasets, we presented a generic concept describing these data representations, that is, " Discretely Sampled Object Representations (DSOR)" [1].

DSOR defines a graphical model using collections of discretely positioned samples. Therefore, it represents certain geometrical or physical properties of sampled objects.

Example Sampling Modality (*physical property*)	**Data Dimensional / Number of Channels**	**Representation Scheme**
Black-white photography (*light reflection*)	2 / 1	2D regular grid
Colour photography (*light reflection*)	2 / 3	2D regular grid
Raw laser scans (*distance to a plane*)	2.5 / 1	2D regular grid
Circular full-body scan (*distance to an axis*)	2.5 / 1	2D curvilinear grid
Computer tomography (*X-ray attenuation*)	3 / 1	3D regular grid
Magnetic resonance imaging (*relaxation of magnetized nuclei*)	3 / 1	3D regular grid
Raw 3D Ultrasonography (*sonic reflection*)	2.5 / 1	unstructured regular grid
Processed 3D Ultrasonography (*sonic reflection*)	3 / 1	3D regular grid
Electron microscopy (*electron diffraction*)	3 / 1	3D regular grid
Spatial distance fields (*distance to surface*)	3 / 1	3D regular grid
Spatial vector fields (*e.g., velocity*)	3 / 3	3D regular grid
3D photographic imaging (*light reflection*)	3 / 3	3D regular grid
Movies and videos (*time-varying light reflection*)	3 / 3	3D regular grid
Particle simulation results (*space-time position, etc.*)	4 / 1	time-series, 3D point set
Motion capture data (*space-time position*)	4 / 1	time-series, 3D point set
Seismic measurements (*space-time density, temperature, etc.*)	4 / n	time-series, 2D point set

Table 2.1: Example data capture modalities, and their typical characteristics and representation schemes [1].

Unlike commonly used data structures in computer graphics, DSORs lack geometrical, topological and semantic information. Hence they pose significant challenges to develop texture mapping systems that directly operate on them.

Table 2.1 lists the major digitisation techniques widely used for acquiring DSORs of real-life objects. As the literature on texture mapping is dominated by surface-based

modelling and rendering techniques, it is certainly sensible to consider the deformation and texture mapping of discretely sampled object representations in the context of these techniques.

It is worth pointing out that discretely sampled object representations let us reconsider the effectiveness of modelling volume datasets.

This includes moving from grid-based volume graphics to point-based volume graphics and the multi-resolution rendering controls for point-based surfaces, demonstrated by Boubekeur et al. [7]. Therefore we believe manipulating point clouds for annotating volume datasets can become a bridge connecting volume graphics and image-based texture models.

Here, the novel concept of *PBVO: the point-based volume object* [8], are particularly interesting to us in that point clouds can become an effective representation of volume objects, by using the technical framework of Constructive Volume Geometry (CVG).

When a digital object is captured in a DSOR, the much desirable geometrical, topological and semantic information is not available. In other words, we are facing the abstraction of discretely sampled datasets as well as the generality of true-3D representations in the context of computer graphics. *Therefore, the challenge of this thesis is to drive a paradigm shift from surface graphics to volume graphics.*

The research will have several impacts upon computer graphics and its applications. In particular,

- it will challenge the concepts and methodologies of volume graphics by stimulating techniques in the less-developed area of realistic volume rendering;
- it will shape new techniques that can benefit from DSOR based applications such as medical imaging and scientific visualisation.

Now we could easily scan a frog, a tomato, a skull, a human body, a lizard, and so on, using digital imaging techniques. But how do we preserve their realism of appearances?

Cyber artists have already demonstrated their artificial computer graphic characters, with charming, realistic, and lifelike characteristics. Portraits of these characters are so lifelike that it is hard to believe they are just computer generated [9].

So if the virtual characters have already stepped into the cyber world in the 21st century, why do volume objects still live in the stone age?

Hence we focus the issues of the realistic appearance of these DSOR volume datasets, in particular:

- image based texture models for volume datasets and highly detailed close-ups.
- lifelike applications in medical imaging and scientific visualisation.

Famous movies providing extremely realistic (life-like synthetic characters) visual effects include: The Matrix: Reloaded, the Matrix: Revolution, The Lord of the Rings: The Return of the King, The Hulk, etc. Twenty categories of visual effects which are widely used in creative fields were introduced by the Visual Effects Society (VES, http://www.visualeffectssociety.com/) [10]. In comparison to these stunning lifelike CG characters (surface-based mesh models), research on realistically rendering volume objects (characters) is still in its infancy. There are so far only a few pieces of reported research which focus on manipulating the volume Visible Man dataset through skeleton based motion controls [1].

As shown in Figure 2-1, continuous models with snake-boundary representations for image-based characters were presented by Froumentin et al. in [11]. Since traditional interpolation methods will blur the sharpness of edges and degrade the quality of texture features, especially the tiny texture features under warping and morphing operations, their continuous models maintain the sharpness of edges and the smoothness of flat areas to eliminate the noticeable visual flaws while the digital character is under deformation.

Another example is given in Figure 2-2. Here, the fire, the lobster and the human body are all DSOR based volume datasets [1]. These volume objects could be manipulated using spatial transfer functions independently [12, 13]. Unfortunately, the human character was rendered without any realistic appearance.

Under such circumstance, it is highly desirable to push rendering based techniques and visualisation based methods to a practical level. In other words, even without geometrical, topological and semantic information, we would like to recover one critical factor for volume objects: realistic appearance.

(a)Snake model of swan (b)Arbitrary warping (c)Digital composition.

Figure 2-1: 2D DSORs can be deformed using a 2.5D rendering and composition system [11].

The Visible Man was about to barbecue a lobster (left) and the lobster exploded (right)

Figure 2-2: Splitting of a lobster using a block-based approach in conjunction with motion of the Visible Man. Both the lobster and the Visible Man are defined using discrete volume representations and the fire is defined using a procedural volume representation [14]

The core of most mainstream pipelines is surface-based modelling and rendering. Extracting surface models from digital imaging data is in general a complex and often ineffective process. The direct use of such surface based data is usually restricted to traditional texture and environment mapping. In addition, some volume-based techniques are also restricted to modelling and rendering amorphous objects, such as fire and clouds. However, as we will demonstrate in this thesis, volume objects could be presented from 2D imagery data (for external specification, for instance, laser scanning range data) and 3D imagery data (for internal specification, for instance, MRI and CT data), that is, bringing the DSORs together through a novel *coherent modelling scheme*.

2.2 Mesh models for Texturing Volume Objects

Mesh based graphic models are widely used. Some successful mesh based digital models are: the female character "Dawn" presented by NVidia [15], the "dinosaur" characters in "The Lost World: Jurassic Park", the "Gollum" character in "The Lord of The Rings: The Two Towers", and the "Shrek" character in "Shrek", etc. In order to achieve such success, processes of intuition, design, creativity, selection, critique and refinement were extensively gone through. Many advanced modelling techniques were used, for instance, free-form curved surfaces, deformations, physical simulations, surface colouring and reflections, multi-layer models, and so on [10].

Unfortunately, there are not so many techniques for texturing DSOR characters. The common techniques are based on converting the discretely sampled datasets to mesh-based geometrical objects. Such conversion inevitably focuses on the exterior information rather then the interior information found in DSOR objects.

In this section we will briefly review the volume based mesh generating algorithms and their advantages and problems.

2.2.1 Mesh-generating: from voxel to polygon

One way to use a volume model is to choose an iso-surface within the volume, create a mesh to fit and then to texture and render it with a conventional surface renderer. Numerous methods have been developed to abstract surfaces from volume objects. Detailed discussions from the volume visualisation view point are given in Jones' PhD thesis [16] and Satherley's PhD thesis [17]. Here, we focus on two generic categories: *surface tilers* and *surface trackers* [17].

Surface tiling

Surface tiling (iso-surfacing) is to traverse the entire dataset to extract surfaces composed of geometrical primitives. Lorensen and Cline's Marching Cubes is one of the most popular surface tiling techniques [18]. Marching Cubes, as the eponym suggests, is to iteratively place a cube over cells of datasets. Triangulations can be further determined as part of the iso-surface contained in a cell. The positions of the vertices of triangles, which represent the iso-surface, can be interpolated using the relevant vertices of the cube. The surface is thus produced from discretely sampled representations

of scalar fields of volume objects, i.e, $f : \Re^3 \rightarrow \Re^2$.

In recent years, many techniques have been presented to improve Marching Cubes. Examples include the algorithm presented by Lewiner et al. to eliminate both face and cell ambiguity [19], the real-time adaptive iso-surfacing method presented by Heidrich et al. [20] to improve the efficiency of computing, and extended marching cubes presented by Kobbelt et al. [21] to preserve sharp features.

Note that Kobbelt et al. [21] used normal information to preserve sharp features. The crucial step in their algorithms focuses on accurately finding the intersection points of tangent planes of zero-crossing points. This method was further employed by Ju et al. in their dual contouring algorithms [22], which combine the extended marching cubes presented by Kobbelt et al. with the *SurfaceNets* methods presented by Gibson [23]. In particular, Gibson demonstrated that just applying a gradient operator to the grey-scaled volume data does not always provide a good estimate of surface normals [23]. That is, normal directions and magnitude may vary much more than we would expect the surface normals of volume objects to vary. Some normals even point in inverse directions, a case which might potentially lead to edge flips.

Even though the previously mentioned methods can, to some extent, preserve sharp features and prevent holes and cracks, they still suffer from ambiguous cases and may have holes due to *topological errors*. In other words, these methods may introduce another problem of inter-cell dependency [24]. Such inter-cell dependency, which is introduced by edge-flipping operations, may cause the efficiency of the whole algorithms to become lower.

Recently, a notable method to improve marching cube algorithms was presented by Ho et al. [25]. They presented a novel solution to the previously discussed problems of surface extraction from volume data. Given a marching cube, it can be unfolded into six marching squares. Each square is processed independently. The generated segments on these faces are put back to 3D to form components. By doing so, the goal of being adaptive without performing crack patching is achieved. In addition, face ambiguities can be resolved in 2D by resolving the ambiguous faces. Finally, the resulting components are triangulated to generated the iso-surface.

Surface tracking

Surface tracking algorithms [26] focus on two steps: find a seed point on the surface and then track the iso-surface through the volume. The tracked iso-surfaces are con-

verted into cell faces. The final surfaces are represented by a list of connected cells, for instance, connected edges, connected faces and connected vertexes.

Once the iso-surface is abstracted as a mesh model, then it can be rendered by any conventional computer graphic system, like Maya, 3DS, Animo, etc. However, these volume originated surface reconstruction techniques still have some disadvantages, for instance, single surface representation, expensive pre-processing, and cracks in multi-resolution meshes.

Abstracting iso-surfaces from volume objects typically assumes a mesh representation as the final output. However, in the following chapters of this thesis, we use *direct surface rendering (DSR)* to calculate positions of iso-value (level-set), rather than constructing a mesh model. We do not construct any mesh models throughout this thesis.

2.2.2 Holes, cracks, topological errors and simplifications

Cracks, holes, ambiguity, topological errors are common problems of surface extraction techniques. The previous subsection gave a brief review of algorithms for extracting surfaces from volume data. Here, we discuss another category of extracting geometry from DSORs, i.e, surface reconstruction from point data. A detailed discussion will be found in Chen et al. [1].

Many data acquisition techniques (e.g., laser range scanning [27] and alpha matte acquisition [28]) generate highly accurate output in the form of an arbitrary set of points in space. Highly accurate geometric models of complex physical objects could be acquired through such scanning techniques. Where the properties of these points cannot be discerned directly, they must be inferred algorithmically. Because of noise and imperfections introduced in the acquisition stage, holes, cracks and topological errors become much more serious and can not be eliminated using trilinear techniques and enhanced lookup tables. Therefore, such surface reconstructing algorithms list noise tolerance as a priority.

Practical issues relative to these static DSORs based approaches are, for instance, obtaining a mesh which is free of holes and inconsistencies [29] and inter-cell dependency due to topological errors [30]. In Figure 2-3, the left image is the photograph of the head of Michelangelo's David. The right rendered image shows a section of the hair. Because of the scanning inconsistency (self-occlusion: a geometrical incon-

sistency), holes and floating islands exist even after scanning from different viewing directions and positions [31].

Note that in [31], Davis et al. focused on the situation in which holes are too geometrical and topologically complex to be filled using triangulation algorithms. In particular, they pointed out that for multiple boundary holes, topologically inflexible methods may fail to find valid manifold surfaces.

Figure 2-3: Geometrical inconsistency: Holes and floating islands exist while generating mesh models for the discretely sampled David. Holes could be filled using volumetric diffusions presented by Davis et. al in [31].

In the digital Michelangelo project [27, 31], an ideal surface extracting technique was defined as having the following properties:

- geometrical consistency of producing a manifold, non-self-intersecting surface.
- geometrical consistency of tolerance to scanning noise (occlusions, equipment imaging or calibrating errors, etc.) and functional errors (for instance, low object reflectance, extreme specularity and surface scattering, etc.).
- practicability of using all available modelling information.
- scalability to billions of samples.
- topological consistency with multi-boundary components.

A flexible topological noise removal method was presented by Guskov and Wood in [29]. An out-of-core algorithm for isosurface topology simplification was described

by Wood et al. in [32]. As shown in Figure 2-4, topological noise becomes obvious in the progressively close views.

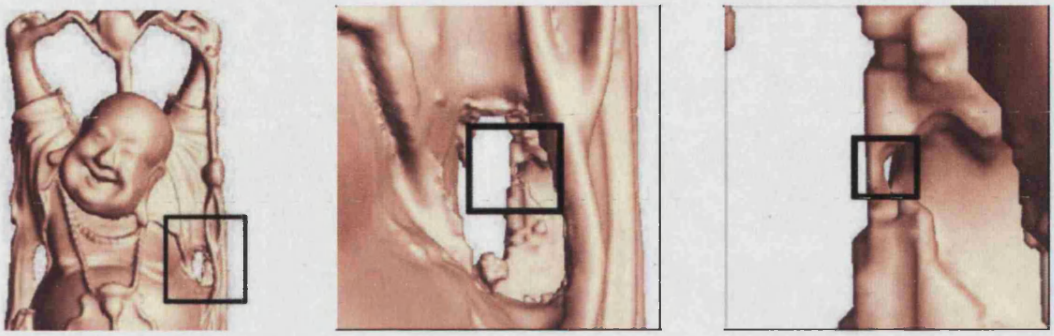


Figure 2-4: Topological inconsistency (an extraneous handle) exists while generating the mesh model for the Budah [32].

Note that the ideal isosurface of the Buddha statue shown in Figure 2-4 has genus 6 [29, 32]. A handle is a toroidal region of the surface with genus 1. (Ref. detailed discussions in [32]). In fact, the real reconstructed surface has genus 104, because of the extraneous topological handles. These topological artifacts become obvious in close-ups.

From the view point of surface reconstruction, these extraneous handles create serious problems for any further geometrical processing, such as mesh simplification and smoothing. Also, these topological artifacts hinder the processing of texture mapping and remeshing. In addition, in medical MRI, topological inconsistency may result in failures in organ fitting, feature registration, and classifications. (Ref. detailed discussions in [32]).

Wood et al. achieved their simplification methods by directly operating on volume representations. They remove topological defects in an isosurface rather then repairing defects on the constructed mesh model [32].

2.3 Smoothing Mesh Models

2.3.1 Mesh inversion

Mesh models such as triangulated surface models and tetrahedral volume mesh models are often used to represent DSOR objects. Moving both the surface mesh model and

the volume mesh model are common operations in surgical planning, crash simulation, and volume animation.

During the operation of mesh moving, the boundary domain of the mesh models as well as the internal vertexes of the mesh models are updated at each time step. However, mesh invertibility is a common problem in mesh moving. In other words, if the element of a mesh model is inverted, then the topological structure of the mesh model will be changed. Holes and cracks might appear.

A good mesh-moving method should avoid any element inversion and preserve the topology of the mesh model. Mesh smoothing techniques such as Laplacian smoothing, Winslow smoothing, mesh refinement, mesh coarsening and mesh enriching, are often used to improve the quality of a mesh during manipulation operations. However, the computation expense and the lack of theoretical guarantees for resisting inversion are the main weak points of these conventional techniques.

Preserving the connectivity of the mesh model is an important feature for annotating volume objects using our image based projective texture models. Our initial targets are realistic texture mapping and high quality close-ups for annotating volume objects. Therefore, the intermediate template should be further smoothed in order to improve the warping quality of texture images. In practice, boundary conditions should be satisfied when using mesh-model based surface flattening methods [33].

We noted that the Linear-Weighted-Laplacian-Smoothing method presented by Shontz and Vavasis [34] addressed the same considerations. The important feature of their methods is that the *connectivity of the mesh model under warping is not changed.* As we will explain in Chapter 5, we will use this in our texture mapping and annotating system.

2.3.2 Laplacian mesh processing/smoothing

Laplacian operations for geometry processing are particularly interesting to us. Linear operators, mesh editing, shape approximation, mesh filtering and morphing, are becoming common techniques [35]. As Sorkine pointed out [35], discretely sampled point clouds that are used to construct mesh models are becoming highly detailed, noisy and complex. The crucial characteristics of a practical mesh operation framework include: (1) detail-preserving operations; (2) linear operations; (3) efficient shape representation.

Using a Laplacian Mesh processing framework, we can represent mesh models using their differential properties, derived from certain linear operators defined on the mesh. These linear operators represent conventional mesh models in the differential based supporting bases, which benefits various manipulation operations.

Differential mesh representations preserve local details such as the size, the orientation and the shape of local geometrical structures.

The linearity of the operations make mesh processing efficient.

The concept of manipulating and modifying a mesh model while preserving the geometric details (connectivity) is also a crucial feature in the application of annotating and texturing volume objects. Given a Laplacian representation of a mesh model, the local differential representations, which are independent of the absolute coordinates of vertexes in Euclidean-space, play key roles in mesh-editing operations and preserve directions of the local geometrical structures.

Several types of local surface representations employing the Laplacian framework have been presented elsewhere, for instance, partially intrinsic surface mesh representations [36] and δ-coordinates based representations with spatial boundary conditions [37].

2.3.3 Geodesic based multi-dimensional-scaling (MDS)

Matching a 2D image onto a flattened object surface could possibly become very tedious under some circumstances. Therefore, Zigelman et. al presented a novel texture mapping method by using Multi-Dimensional-Scaling (MDS) [38]. The advantages of their Geodesic MDS methods are that mapping a set of 3D points on objects into a flat 2D domain. Their method yields minimal changes of the Euclidean distances between the flattened corresponding points.

The key points of their methods are first calculate the geodesic distances for each pair of vertex pairs on the volume object, then using this geodesic distance matrix to flattening the 3D points cloud onto 2D Euclidean plane. The advantages of their methods are of particular interest to us in that, after flattening the 3D points on the surface of the object, the texture image can be possibly overlaid onto the 2D Euclidean plane directly.

Such methods allow us to improve the shear effect of projective mapping by directly manipulating a 3D point cloud by, for example, flattening the 3D point clouds

within the volume object. Note that in their algorithm, the surface model must guarantee the existence of geodesic-paths for each pair of the 3D points.

2.4 Volume Texture Models

2.4.1 Solid textures and other 3D textures

Direct rendering methods which avoid the need for intermediate meshes [6, 39, 40, 41], are with the particular advantage to us.

Winter was the first to adopt Bier and Sloan's two-part texture mapping for texturing volume objects [6]. His work consists of a projective texture method for volume objects. We use this in our approach; in addition, we offer novel semantic coherent models to control texture extruding through the volume and texture smearing over large areas [39, 40]. In this section, we will briefly review the texture mapping techniques in volume graphics. (*For more detailed discussions on volume graphics please refer to the PhD theses of Winter [6], Satherley [17] and Rodgman [42].*)

Image mapping is a recognised shortcut for simulating surface characteristics. It enriches the surface shading process by using images to simulate surface texture and some other important attributes such as roughness and reflectivity (ambient reflection, diffuse reflection, specular reflection, etc.). Some typical image maps include: environment maps, colour maps, procedure maps, bump and displacement maps, transparency maps, etc. Typical image mapping projections include flat projection, cubical projection, spherical projection and warping projection [10].

One research topic of volume graphics is increasing photorealism by introducing surface graphics techniques such as shadows, reflection, refraction and illumination into volume rendering. Volume based non-photorealistic rendering (NPR), hypertexture and bump maps were also developed. Unfortunately, these techniques have yet to produce realistic appearances for volume objects. Textures of real physical objects around us depend not only on colours and physical characteristics but also on human perceptual and cognitive factors.

One advantage of volume graphics over surface graphics is that texturing can be performed on a volumetric object as a preprocess prior to rendering, rather than as the final matting process in surface graphics [17]. Therefore, successful applications of amorphous effects like fire, hair, glow and melting have been demonstrated using solid

textures, procedure textures, bump maps and hyptertextures [6, 14, 43].

Solid texture was presented by Perlin [44] and Peachey [45] independently in 1985. It is designed for overcoming the limitations of parametric texture-mapping techniques, for complex surfaces which may not have parametric forms. Only a location in space of the surface point needs to be determined for each screen pixel onto which the surface projects. A texture value is established by evaluating a procedural spatial texture function. This eliminates surface colour discontinuities because of poorly-drawn textures or poorly-defined surface parameters. Procedural solid textures by definition have infinite precision and are, therefore, suitable for close-up viewing. Note that, even though procedure texture by definition has infinite detail, it does suffer from "minification aliasing" (several texture elements are projected onto the same screen pixel) if texture frequencies are above the Nyquist limit of the texture raster [47]

In its basic form solid texture is a function of space, not of the object. Addressing this, Carr, Hart and Maillot [48] described the solid map method. This generates a 2D texture map from the solid texture, for each polygon of the model. These textures are then glued to the polygons in the usual way, with the advantage that the textures move and distort in the same way that the surface does. This is a major advantage for animated, flexing objects. Moreover the texture is no longer determined at screen resolution, with significant gains for image quality control. This was further investigated and presented as the meshed atlas method by Carr and Hart [49]. One attraction of their approach for us is that it combines the infinite detail of procedural texture with the generality of 2D texture image mapping.

Hypertexture is also a space function, one which manipulates the 3D regions around surfaces of volume objects. Given an iso-surface of an object, the 3D space is represented by three states, inside the region (represented by a constant scalar value), outside the region (soft region, modeled by a scalar value function) and the boundary (conventionally assigned a value of zero). Hypertexture effects can be achieved by manipulating density modulation functions (DMF) in the soft regions, $D(p)$, through the following repeating operations:

$H(D(p), p) = DMF_n(...(DMF_0(D(p))))$

where p represents the 3D position within the soft region. As Miller and Jones suggested [43], distance field volume representations, hypertextures, bump maps and procedural texture techniques can be flexibly combined together to construct an efficient GPU based renderer.

Owada et al.'s work on texturing volumes [41] concentrates on static illustrative images, with the "interior" texture being applied to the cross-sections. In their method, internal textures can be browsed by cutting the illustrative mesh models at desired positions. In particular, the designer must provide guiding information to set up the correspondence between the cross-section and a reference 2D image of internal textures. Such guide information is stored with the mesh model of the target object. The system can thereafter synthesise internal textures which can be visualised on any cross-section. There is no need to maintain any volumetric data. Therefore, their technical contributions are: first, the interfaces that are used to assign textures to a given surface mesh; second, the algorithms that synthesise textures on cross-sections using 2D images. Their system could enrich human communication in the areas such as medicine, biology and geology, etc.

Their method does not readily generalise to dynamically-varying cross-sections. In fact, providing guiding information throughout 3D space might be hypothetically similar to the scalar field based volume modelling techniques described by Winter [50] using lathe and sweep control.

Owada's methods focus on providing realistic internal textures of mesh-based objects using illustrative images. Similarly, Winter's algorithms offer heterogeneous interiors and amorphous effects from swept volume objects. The importance of constructing image based 3D textures and image based volume objects is obvious.

A generic model of image-swept volumes was introduced by Winter and Chen in [50]. Their 3D trajectory function, $a(u)$, can be extended to other spatial functions, for instance the projection functions widely used in texture mapping [10]. As we will demonstrate in the next chapter, projective texture models can thus be constructed in the same manner by stacking pixels in 2D image templates onto the voxels swept by 3D indexing functions.

The concept of an image-swept volume can be described [50] as: normalising a given image, $n_x \times n_y$, onto the sweeping template: $m(x, y), 0 \leq x, y \leq 1$. The 3D trajectory function can be defined as:

$$a(u) = (a_x(u), a_y(u), a_z(u)), u_{min} \leq u \leq u_{max}$$

where $(a_x(u), a_y(u), a_z(u))$ give the 3D positions running along the trajectory. Assume $r(x, y, u)$ is the affine transformation which changes the orientation and size of

the template $m(x, y)$ along the trajectory $a(u)$. The swept volume can be defined as $\Gamma = \{\gamma_1, \gamma_2, ...\}$. This model contains instances of mapping points in a sweeping template to positions in the volume. The instance can be defined as: $\gamma_i =< p_i, v_i >$, where p_i is the point in sweeping template, v_i is the position in volume. Integrating the transformation function r, the instance function can be defined as:

$$\gamma(x, y, u) =< r(x, y, u) + a(u), m(x, y) >$$

So given a point $m(x, y)$ and a 3D trajectory position $a(u)$, an instance can be generated by shifting the transformed point $r(x, y, u)$ onto its new 3D location $r(x, y, u) + a(u)$. By substituting 3D trajectory functions with other indexing functions and by adapting other transformation functions, we can extend the above sweep models to a more generic *image based indexing volume (IMIV) model* .

Both Owada's and Winter's techniques acknowledge the importance of 2D images by using them as original texture templates to construct the 3D contents of volume objects. In fact, from their papers we can conclude that 2D texture images produce some stunning images of the constructive volume objects. Therefore, there is no doubt that textures from 2D images can be sufficient, consistent and practical to reproduce the lifelike appearances of physical objects for virtual DSOR characters.

Texture synthesis models construct 2D or 3D textures which are similar to their reference images. A detailed discussion on texture synthesis was presented by Wei in his PhD thesis [51]. Four categories of 3D texture synthesis algorithms are: frequency domain [52], pixel-based [51, 53], patch-based [54] and non-periodic tiling and sampling [55].

The need for a high-quality and realistic appearance of a volume object was also addressed by Wang and Mueller [56]. They use constrained texture synthesis to extend image-guided detail enhancement to multiple levels of scale. New detail is synthesised to match the local data, scaled appropriately. They demonstrate this both in 2D (a "virtual microscope") and in 3D, for volumetric viewing. They use the term *semantic zooming* to express this; that is, "each level of detail stems from a data source attuned to that resolution".

For our interests, colour information in illustrative images could also be transferred to voxels in volume objects [57, 58, 59]. As Lu and Ebert demonstrated in [57], volume illustration can have more clearly delineated objects, enriched details, and

artistically visualised volume objects. Colours are transferred based on the clustering and similarities in the example (illustration) images and volume objects.

As previously mentioned, Winter used instance functions $\gamma(x, y, u)$ to construct swept volume objects from image template $m(x, y)$ [50]. Colour information and intensity (geometrical) information are constructed in this way. In fact, Lu and Ebert's technique, which uses clustering, mapping and transfer methods to construct the colour indexing function, can be used as an optional candidate to instance functions in Winter's swept volume model.

It seems that research has shifted from 2D texture images to image-based volumetric textures. As Sousa et al. offered in [58], their *emphatic* rendering system can produce images that simulate pictorial representations, for both scientific and biomedical visualisation. Their system combines traditional and novel illustration techniques, to guide and facilitate the learning of complex biomedical information, i.e, structural, functional and procedural information.

2.4.2 Projective texture models

A texture mapping process which does not require uv-parameterization of the recipient surface was presented by Bier and Sloan [60] in 1986. The texture is first mapped onto a plenoptic intermediate surface (the *S*-mapping) and is then projected onto the object (the *O*-mapping). Intermediate surfaces tend to be simple geometrical primitives such as a cube, sphere or cylinder.

When we use direct volume rendering of a volume dataset, there is no surface onto which to map texture directly. However we can use two-part texture mapping and we will make use of this in our method. To do this will require the inverse functions of the *S*-mapping and *O*-mapping.

Environment mapping [61] simplifies ray-trace rendering by treating the environment as a 2D projected image, which is typically represented on a spherical, cylindrical or cubic plenoptic surface. This is then treated as the illumination of the object contained within the plenoptic surface. This idea of mapping the world onto a surface has other uses and we will use it in our method.

2.4.3 Annotating volume data

Annotating volume objects is an important task in visualisation and surgical planning. As Kniss et al. [62] and Tzeng et al. [63] demonstrated, multi-dimensional transfer functions split the 3D space into different semantic layers and thus segment and classify volume objects effectively, for example by extracting specific material boundaries and conveying subtle surface properties. We see that, with the viewpoint of volume rendering based texture mapping, transfer functions not only help us to classify the volume object into different layers but also tell us the locations of where appropriate textures could be applied. Therefore, transfer functions can be used during the volume rendering to manipulate and guide texture placement. As we demonstrate in this chapter, our 2.5D texture engine is constructed by using three components: projective texture mapping, transfer functions and volume rendering.

2.4.4 Interpolation methods

We note that image-based texture maps have their advantages. They do not require a procedural definition but only expect the user to provide a pixel map containing the texture. This supports a very general class of textures but always at a fixed resolution. There are techniques for enhancing the resolution of an image, so there is promise in using a texture map for its generality while interpolating higher-resolution texture on demand.

Candidate interpolation techniques include bilinear, bicubic or cubic B-spline interpolation; and feature-based methods such as edge directed interpolation (EDI) [64] and the new edge directed interpolation (NEDI) [65]. These could be used to regenerate the texture image with high resolution. Su and Willis [66, 67] describe a fast method which takes edge information into account, in order to retain visual sharpness when the image resolution is increased.

2.5 Multi-dimensional Transfer Functions and Pseudo-Colour Information

Realistic appearances of volume objects become more and more important for surgical planning and entertainment. The motivation of illustrative visualisation was described

as: "An illustration is a visualisation such as drawing, painting, photograph or other work of art that stresses subject more than form. The aim of an illustration is to elucidate or decorate a story, poem or piece of texture information (such as a newspaper article) by providing a visual representation of something described in the text" (Wikipedia, [68]).

Here, techniques depicting and illustrating isosurfaces, particular those which do not extract explicit surface geometry (meshes or functional models), are particularly interesting to us. These techniques were implemented on the basis of direct volume rendering and direct voxel manipulation, that is, they do not construct mesh models as intermediate steps. Therefore, there is no worry about the artifacts of mesh surface extracting algorithms described in the above sections.

Detailed discussions on illustrative visualisation [69] include: human visual perception and illustrative aspects of art, illustrative and non-photorealistic rendering, illustrative visualisation for isosurfaces and volumes, smart visibility in visualisation, interactive volume illustration for medical and surgical training, and illustrative visualisation for surgical planning.

Extremely large datsets and multi-channel visualisations, which lack sufficient geometrical, topological, and semantic information, are typical challenges in medical and entertainment visualisations, where it is necessary to convey real-time manipulability to end users. Some practical annotation techniques are discussed in the following subsections.

2.5.1 Deferred shading

Common isosurface rendering techniques in the volume rendering community are ray-casting methods, which are based on ray-isosurface intersections [70, 71, 72]. Shading calculations in these algorithms are conventionally executed on all voxels. Because of the complexity of shading computation, real-time visualisation applications can become impractical. Therefore, the *Deferred Shading* technique, shading only for visible voxels/pixels, was described by Deering et al. [73], Lastra et al. [74], and Saito and Takahashi [75], to improve computing efficiency. Using a floating point image which saves the ray-isosurface intersections as its input, the technique thus reduces its computing complexity from voxels in volume spaces to pixels in the final output.

Deferred shading techniques provide two crucial advantages for manipulating vol-

ume data: first, the effective representation of volume spaces, i.e, the visibility information of voxels during rendering; second, differential information, colour mapping of curvature magnitudes, flow advection along curvature directions, and solid textures, can be applied to isosurfaces. Curvature magnitudes were visualised by mapping them to colours (*not textures*) via one- or two-dimensional transfer functions. Surface structures such as ridge, valley lines and silhouettes can be rendered and annotated accordingly.

2.5.2 Multi-dimensional transfer functions and hierarchical transfer functions

Transfer functions are designed for classifying datasets during volume rendering. Optical properties like colour, opacity, refraction and reflection coefficients can be assigned to the *classified* values the datasets consists of.

Hladuvka et al. [76] define a transfer function as:

$$F_1 \times F_2 \times ... \times F_n \longrightarrow O_1 \times O_2 \times ... \times O_m \tag{2.1}$$

where $F_i, i = 1, 2, ..., n$ are scalar fields, $O_i, i = 1, 2, ..., m$ are optical properties. The above transfer function is therefore defined as a mapping from a cartesian product of scalar fields F to a cartesian product of optical properties O.

It is not an easy task for end users to develop useful transfer functions for their applications. Discussion of generic transfer functions assigning opacity, colour and emittance properties were offered by Lichtenbelt et al. [77]. The extended categories of transfer functions include: optical model [78], first and second derivative model [79], image model [80], topological model [81], interactive multi-dimensional model [82], banded widget model and vector calculus operators (i.e, gradient magnitude $|\nabla U|$, the Laplacian $\nabla^2 U$, vector magnitude $|U|$, and curvature $|\nabla \times U|$) for flow visualisation [59].

The geometrical structures of iso-surfaces were mainly represented by gradient and curvature information, for instance, principal curvature magnitudes k_1, k_2, and gradient vector $g = (g_x, g_y, g_z)$. Colour mappings of mean principal curvature magnitudes $(k_1 + k_2)/2$ and Gaussian curvature $k_1 * k_2$ can be constructed via a 1D colour lookup table. Colour mappings of other geometrical structures on iso-surfaces can be calculated using two-dimensional curvature magnitude transfer functions, (k_1, k_2). Typical

applications of curvature-based transfer functions can be seen in [76, 83].

The transfer function mapping principal curvature to colour and opacity can be defined as [76]:

$\tau : k_1 \times k_2 \longrightarrow R \times G \times B \times \alpha$

Figure 2-7 shows an analytic example of a curvature-based transfer function given by Hladuvka et al. [76]. The *pseudo-colours* in the domain of the two-dimensional transfer function are transferred to the 3D scalar values according to their principal curvatures. Such a definition is useful to distinguish among shapes, set smooth transitions, and set transfer functions.

Some volume rendering techniques, (for instance, texture splats for 3D scalar and vector field visualisation presented by Crawfis [84], and flow volumes for interactive vector field visualisation presented by Max et al. in [85]), cannot sharply focus on interesting features while still representing the three-dimensional structure [59, 86]. Svakhine et al. offered illustrative style transfer functions, extended 2D transfer function widgets and new banding transfer function widgets, to enhance the accuracy and perceptibility of their flow visualisation system [59]. Here, we refer to hierarchical transfer functions as multi transfer functions which could be iteratively applied to different materials during rendering.

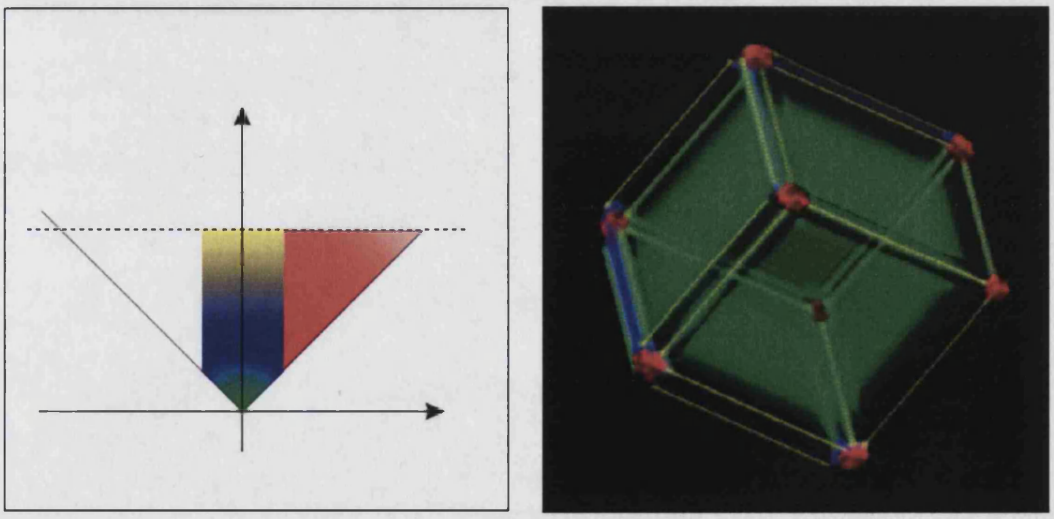


Figure 2-5: Principal curvature based colour transfer function (left) and the rendered cube [76].

2.5.3 Illustrative enhancements

Illustrative enhancements include *boundary enhancement* (boundaries between materials are areas with high gradient), *silhouette enhancement* (modifying the opacity according to the dot product of view-vector and gradient vector), *sketching* (if the silhouette term is small, then it is assigned with lower opacity), edge colouring (modifying colours rather than opacity), tone shading and illumination, distance colour blending and feature halos. Detailed discussions on interactive volume illustration for medical and surgical training were given by Ebert [87].

Some other illustrative enhancement techniques include experimental advection, photographic flow visualisation, advanced two-dimensional functions, focal and contextual illustrative styles, novel oriented structure enhancement techniques, to allow interactive visualisation, exploration, and comparative analysis of time-varying volume datsets [59]. *Interactivity* and *flexibility* are the key features of a useful visualisation system.

2.5.4 Scalar fields and field transfer functions

Scalar fields and field based transfer functions are popular and effective techniques used in volume visualisation and annotation. Detailed discussions on scalar field models of volume objects were offered in [1, 6, 14, 16, 17, 42]. A system level discussion about scalar fields and their operations was addressed by Chen and Tucker [88]:

Let $\mathbb{R}$ denote the set of all real numbers, and $\mathbb{E}^3$ denote 3D Euclidean space. A *scalar field* is a function:

$$F : \mathbb{E}^3 \longrightarrow \mathbb{R} \tag{2.2}$$

The tuple of scalar fields defined in $\mathbb{E}^3$ is $\mathbf{o} = (O, F_1, F_2, ..., F_n)$. In particular, opacity field $O : \mathbb{E}^3 \longrightarrow [0\ 1]$, specifies the "visibility" of every point p in $\mathbb{E}^3$. Other attribute fields $F_1, F_2, ..., F_n$ could be optical or geometrical properties of volume objects, for instance, colour components (red, green, blue, or luminance and chrominance components), refraction coefficients, reflection coefficients, specular coefficients, ambient coefficients, diffuse coefficients, specular exponent coefficients, intensities, gradients, principal curvatures, and some non-graphical properties like magnetic field and distance field, etc.

A field could be built from one or more other fields using appropriate mapping functions (as offered by Islam et al. in [14]). Operations on scalar components can be linear, nonlinear, vector or tensor based operators.

Given a finite set $P = \{p_1, p_2, ...p_n, |p_i \in \mathbb{E}^3\}$ of distinct points, the convex hull $\Upsilon(P)$ of the point sets P is the volume of P. $p_1, p_2, ...p_n$ are voxels in this volume. When each voxel p_i is associated with a scalar value v_i, and the value at every other voxels in the convex hull $\Upsilon(P)$ is calculated using an interpolation function I, then we could define an interpolated scalar field F by:

$$F(p) = I(p, (p_1, v1), ..., (p_n, v_n)), p \in \Upsilon(P) \tag{2.3}$$

Interpolation functions such as trilinear functions for regular grid volume datasets, and barycentric interpolations for non-regular (3D tetrahedralisation grid) are commonly used. Finally, given an interpolated scalar field in volume visualisation, we could obtain the necessary opacity and optical property fields using *mapping* functions:

$$O(p) = M(I(p, (p_1, v1), ..., (p_n, v_n))), p \in \Upsilon(P) \tag{2.4}$$

$$F_i(p) = M_i(I(p, (p_1, v1), ..., (p_n, v_n))), p \in \Upsilon(P) \tag{2.5}$$

where M are opacity mapping functions and M_i are other scalar fields mapping functions. Operations on the tuple $\mathbf{o} = (O, F_1, F_2, ..., F_n)$ can be generally represented by *field transfer functions* (Equation (3.19)), and be effectively modelled by *constructive volume geometry* operations [88].

2.6 Semantic Constraints: Splitting and Multi-level Volume Rendering.

Although DSORs may capture a collection of objects in a scene, they do not normally contain any semantic information about the objects of interest, such as object identification and object hierarchy. Tagged volume objects and their segmentation masks, i.e., the classified volume identification indexes (IDs), are necessary information which must be provided for using hierarchy transfer functions.

Without segmentation and the associated semantic information provided by segmentation, any practical applications of visualisation, such as medical training and surgical planning, will be impossible [1].

2.6.1 Splitting and segmentation

A notable *attribute-based spatial and temporal splitting model* was presented by Islam et al. [14].

The concept is more suitable for volume data processing, i.e., segmenting, splitting, deforming, manipulating, animating, annotating, illustrating, and, of course, visualising volume datasets. With such generality, it is easy and flexible to construct more advanced and complex hierarchical visualisation systems.

As we will demonstrate later in this thesis, our semantic projective model is an extension and a novel contribution to Islam's spatial and temporal model.

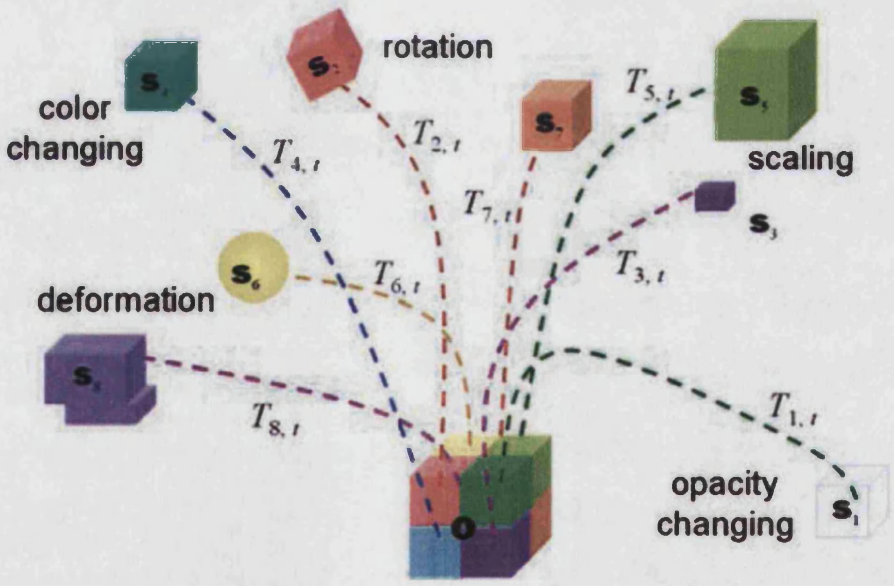


Figure 2-6: An action of splitting a spatial object [14].

Chen et al. [89] developed the concept of *Spatial Transfer Functions (STF)* which enable deformation defined as volume objects in a volume scene graph. Islam et al. [14] further developed this model by incorporating volume splitting operations,

to facilitate the spatial and temporal representations of hierarchical volumetric spaces, i.e., tagged volume spaces with split IDs.

As shown in Figure 2-6, splitting a volume dataset in the spatial and temporal domains could be defined following Islam et al. [14] as follows:
Given an arbitrary spatial object $\mathbf{o} = (O, F_1, F_2, ..., F_n)$ and a set of purposely constructed component objects $s_1, s_2, ..., s_k$, the splitting action is a series of transformation functions $T_{i,t}$ applied to the constructed component s_i, $i = 1, 2, ..., k$, over a period of time $t = 0, 1, ..., t_{max}$. O is the opacity field which describes the visibility of volume objects, $F_1, F_2, ..., F_n$ are scalar fields describing the other properties of volume objects. At $t = 0$, the union of all transformed objects $\mathbf{s}_i = T_{i,0}(s_i)$ is equal to the original space object $\mathbf{o}$. That is:

$$Specification: \ \mathbf{o} = \bigcup(T_{1,0}(s_1), T_{2,0}(s_2), ...T_{k,0}(s_k)) \tag{2.6}$$

$$Transformation: \ \mathbf{O}_t = \bigcup(T_{1,t}(s_1), T_{2,t}(s_2), ...T_{k,t}(s_k)) \tag{2.7}$$

$$Transformation: \ \mathbf{O}_t = \bigcup(O_{1,t}, O_{2,t}, ...O_{k,t}) \tag{2.8}$$

where $\bigcup$ is the constructive volume geometry union operation [14, 88]. $O_{i,t}$ is the transformed split components. Specification operations split the spatial object $\mathbf{o}$ into a set of components, $\mathfrak{F} = \{s_1, s_2, ..., s_k\}$, at initial status $t = 0$. Conventional splitting functions $T_{i,0}$ are geometrical splitting functions, logical splitting functions and spatial transfer functions. The transformation functions $T_{i,t}$ are deformation functions, animation functions, field functions, spatial or temporal transformations, and so on.

2.6.2 Generic volume model, two-level rendering and smart visibility

Equations (2.6) to (2.8) offer us the concept of a *generic volume model*: First, volume objects could be specified as a set of tagged components. Second, each component can be rendered with its own transformation functions.

Two-level volume rendering [90], is composed of a global rendering mode and a local object-focus rendering mode. In other words, a single ray could pierce different tagged objects which have their own illustration configurations.

Smart visibility visualisation [69], is driven by feature importance information. Dynamic changes in visualisation could be rendered using cut-away or ghost-viewing techniques. The spatial arrangement of structures is modified. Leafing, peeling, spreading, and splitting are often used to separate context information. Using smart visibility visualisation techniques, the *inner focus* information can be exposed and visualised accordingly.

The integration of multi visualisation techniques includes: two-level volume rendering techniques [91], smart visualisation techniques [92], tagged ID volume representations [90, 93], global and local compositing buffers [90], and so on.

2.7 New Challenges in Volume Visualisation: Perception based Evaluation.

"Culture is the epidemiology of mental representations: the spread of ideas and practices from person to person" - Dan Sperber, Cultural anthropologist [58, 94]

We argue that *visualisation* is not mature enough to address the problem of effectively spreading ideas. However, as Ebert stated [87], "Visualisation is most powerful when combined with: effective enhancement and extraction of information, perception research, advanced illumination and shading, art and illustration techniques, improved interaction, and a large solution".

Although equations (2.6) to (2.8) offer us a generic model for volume visualisation, the perceptual and cognitive communications between human and rendered information are not covered. In this regard, equation (2.8) could be modified to:

$$PerceptualRepresentation: \ \mathbf{V}_t = PCF_t(O_{1,t}, O_{2,t}, ...O_{k,t}) \qquad (2.9)$$

where PCF_t is the *perceptual and cognitive function (PCF)* operating on the transformed components at time t, and V_t is the neuropsychical model *reconstructed* and *reinterpreted* by the human brain.

Over the past few years, the importance of human perception based rendering techniques has been recognised by the visualisation community [2, 95]. Human factors, such as lighting configurations, visual accuracy, surrounding items, colour scales, were investigated to improve the effectiveness, efficiency and intuitiveness of volume visu-

alisation systems [96, 97, 98].

2.7.1 Quantity and effectiveness

The urgency of investigating the quantity and effectiveness in volume visualisation comes from the real failing moment when "neurosurgeons and radiologists used one of volume renders of the brain and cerebral vasculature during their surgical planning for a patient" [99]. In fact, we still cannot completely and effectively evaluate a visualisation system for its target users [100]. This has drawn the attention of the visualisation community. The current research interests and efforts are shown in *National Institutes of Health (NIH) and National Science Foundation (NSF) Visualisation Research Challenges* [101].

Current evaluation techniques include: expert reviews and experiences [2], user study guidance: "why, how, and when" [102], point-based surface uncertainty [103], and visualisation errors [99]. It is shown by Johnson in his survey [99] that visualisation errors and uncertainty can come from:

- Acquisition: instrument measurement error, numerical analysis error, statistical variation.
- Model: both mathematical and geometric.
- Transformation: errors introduced from resampling, filtering, quantisation, and rescaling.
- Visualisation.

Further research on effectively evaluating visualisation techniques would benefit the following areas which are highly relevant to the aim of this thesis, particularly from the viewpoint of preserving realism for volume objects:

- Modification to data and/or visualisation attributes.
- Improvement to psycho-visual metaphors.
- Better use of annotation and interactive information overloading.

2.7.2 Convince and confidence

By highlighting significant features and subjugating less important details, volume illustrations (NPR plus volume visualisation) can help users perceive communicative information effectively. However, end users, such as doctors and medical students still do not have confidence to use NPR based illustration systems, due to the drawing style of rendered images and possible loss of crucial details. The official validation of illustration systems has yet to be approved [100, 101, 104].

Many subsidiary details cannot be neglected since they play crucial roles in human perceptual and cognitive process. The key point is to effectively preserve these sensitive textures for volume data rather than subjugate these tiny details.

As we will argue in this thesis, image based annotated models could preserve such crucial information in segmented volume objects, whereas, conventional illustration and annotation techniques can not.

2.7.3 Optimal and intelligent visualisation

In practice, end users of visualisation systems need to make extreme effort to explore volume data sets to obtain the best viewpoint and gain the most intuitive visual impressions. Here, intuitiveness can be defined as "immediate apprehension and cognition" [95]. To improve the effectiveness of volume visualisation, optimal viewpoint descriptors have been developed such as surface area entropy, curvature entropy, silhouette length, silhouette entropy, etc. [105]. Volume based optimal view descriptors were developed, such as the feature-driven approach, and view goodness, view likelihood and view stability criteria [106, 107].

Most of the visualisation techniques, for instance, transfer functions, illustrations, annotations, segmentations, classifications, registrations, and so on, involve the user's interactions to guide visualisation processes [101]. Unfortunately, end users often lack such expert knowledge.

AI techniques such as machine learning based SVM (Support Vector Machine) [108] and clustering algorithm ISODATA (Interactive Self-Organising Data Analysis Technique) [109], demonstrate the communicative capabilities of intelligent visualisation techniques by providing effective and intuitive interactions between human and computers.

An intuitive and intelligent user interface was offered by Tzeng and Ma [109].

In their system, object classes are constructed by using the ISODATA method which derived from K-mean clustering algorithms, and are represented in a set of visualisation widgets. In particular, their system supports complex manipulation operations on different clusters, i.e., splitting, merging and discarding clusters.

2.8 Conclusion

In this chapter we described the challenges and the background of volume visualisation and texturing mapping. We observed that the study of preserving the *realism* of volume objects is still in its infancy. We showed that our observations coincide with the current attentions of visualisation communication, described in"NIH/NSF Visualisation Research Challenges Report - January 2006" [104]. We also discussed common perception and cognition related factors for visualising volume objects.

The challenges and the observations motivate us to transfer the realism of images onto volume objects to recover and preserve texture details which are subjugated in conventional illustration processes, as realistically as we can.

We described the discretely sampled object representations (DSORs). We discussed the problems, such as cracks and holes, of traditional mesh abstracting techniques. We justified the properties of different 3D textures, such as solid textures, hypertextures, and two-part texture mapping.

In addition, the comparison among segmenting, splitting, and general modelling volume objects was made. Discussions about multi-dimension transfer functions, semantic constraints, generic volume model, two-level rendering, and smart visibility, were given. We concluded that current illustration techniques might be powerful for illustration and visualisation, but not realistic enough, particulary for volume objects using pseudo-colours and subjugating sensitive textures.

In the next chapter, we will introduce projective texture models. The solutions to the problems such as *volume self-occlusions*, *texture penetration*, *multi-resolution representation* and *colour transfer* will be discussed in the subsequent chapters (Chapters 4, 5, and 6).

Chapter 3

Projective Texture Models

We recognise that volume datasets are an increasingly rich source of material and offer opportunities not present in the traditional surface models widely used. However, by comparison they present a number of challenges. Preserving appearance realism is one of these and is the subject of this chapter.

By working with iso-surfaces (level sets) within volume data, rather than extracting surfaces as meshes, there are particular benefits that we can exploit. In this chapter we will explain:

- A three-part texturing method, which allows us to match the texture to the shape of the object. We use hierarchy volume rendering for the intermediate template needed in our method, which means we can have different templates for different layers within the volume; and we can also have different layers of the volume on the same template. This gives the great flexibility in mapping texture to the volume data and more options at render time than with surface texture mapping.

- A simulated continuous texture from a discrete one, which permits high accuracy of positioning and, importantly, high quality of rendering even in close-ups.

- A method of extruding texture through the volume, which permits direct sculpting and cutting of the volume data, without needing to re-map the texture. This permits interactive sculpting with the texture in place. This applies whether we start with captured volume data or computer-generated volumes and is especially useful with constructive volume geometry (CVG) [88].

- A method of transferring colour from illustrative images to volume data. This

supports constructive volume graphics (CVG) operations and advanced transfer functions. Colour transfer functions are constructed through cluster matching, statistical colour correction, and field functions. Colour realism in illustrative images could be transferred to volume datasets, whereas, pseudo-colours in NPR techniques are hypothetically assigned to voxels through transfer functions for the purpose of communication only.

3.1 Introduction

Validation, *realism* and *affordability* are three key criteria for surgical visualisation and medical virtual environments [100]. In this chapter, we focus on offering *realism* to volume objects, by integrating the *generic volume object model* and hierarchical transfer functions.

We present an approach to texture mapping volume datasets. The approach is based on multiple constraints and continuous space mappings to ensure good image quality. It requires only one intervention by the user, to determine key points where the texture must match an intermediate image of the original data. This can also be used to avoid the problem of texture being smeared over too large an area.

The method is composed of three parts: *semantically generating intermediate templates*, *selectively forward and inverse indexing*, and *volume rendering*. This three-part aspect additionally allows the texture image to be independent of the volume data. We demonstrate an extension to 2.5D textures, extruded through the volume, using an approach consistent with 2D texture. A data-dependent triangulation method is used to retain edge quality in texture images. In addition, colour transferring techniques, which transfer colour information of pixels in illustrative images to colour fields of voxels in 3D space, are presented. Finally, we discuss the challenges of our projective texture model such as texture smearing, penetrating and self-occlusion.

3.1.1 Generic volume model and transfer functions

Starting from Chen's spatial functions [89] and Islam's splitting functions [14], as previously discussed in Subsection 2.5, the generic volume model is defined by one *Specification* (Equation (2.6)) and two *Transformations* (Equations (2.7) (2.8)) as follows:

Given an arbitrary spatial object $\mathbf{o} = (O, F_1, F_2, ..., F_n)$ and a set of purposely constructed component objects $s_1, s_2, ..., s_k$, the splitting action is a series of transformation functions $T_{i,t}$ applied to the constructed component s_i, $i = 1, 2, ..., k$, over a period of time $t = 0, 1, ..., t_{max}$. O is the opacity field which describes the visibility of volume objects, $F_1, F_2, ..., F_n$ are scalar fields describing the other properties of volume objects. At $t = 0$, the union of all transformed objects $\mathbf{s}_i = T_{i,0}(s_i)$ is equal to the original space object $\mathbf{o}$. That is:

$$Specification: \ \mathbf{o} = \bigcup(T_{1,0}(s_1), T_{2,0}(s_2), ...T_{k,0}(s_k))$$

$$Transformation: \ \mathbf{O}_t = \bigcup(T_{1,t}(s_1), T_{2,t}(s_2), ...T_{k,t}(s_k))$$

$$Transformation: \ \mathbf{O}_t = \bigcup(O_{1,t}, O_{2,t}, ...O_{k,t})$$

In the above generic volume model, $O_{i,t}$ is the transformed split component. Specification operations split the spatial object $\mathbf{o}$ into a set of component, $\mathfrak{F} = \{s_1, s_2, ..., s_k\}$, at initial status $t = 0$. Splitting functions $T_{i,0}$ are geometrical splitting functions, logical splitting functions and spatial transfer functions. The transformation functions $T_{i,t}$, $O_{i,t} = T_{i,t}(s_i)$, are deformation functions, animation functions, field functions, spatial or temporal transformations, and so on. $\bigcup$ is the set of CVG operators [88].

Therefore, the *generic volume model* is defined as a DSOR object $\mathbf{o}$ combining with its dynamic behaviour $T_{i,t}$.

Constructing effective splitting functions and transformation functions in volume visualisation applications becomes a primary and important task. Therefore, segmentation techniques are often used to abstract classification information to identify different parts of volume space.

Transformation functions could be extended to a generic description, that is: *semantic functions*, which operate on space criteria, logical criteria, temporal criteria, geometrical criteria, topological criteria, and so on. Starting from transformation functions $T_{i,t}$, and following the spatial definitions given in [89], the *semantic transfer function* could be defined as:

- **Transfer Functions**

Given a spatial object $\mathbf{o}$:

$$\mathbf{o} = (F_0(p), F_1(p), F_2(p), ..., F_n(p)) \tag{3.1}$$

where $F_i(p)$ is either the scalar field defined on 3D position $p \in \mathbb{E}^3$ or the attribute field defined in $\mathbb{E}^3$. With Equation (3.1), not only scalar fields such as geometrical structures

and physical properties, but also attribute fields such as logical constraints and tagged volume IDs, are defined at every point p in $\mathbb{E}^3$.

A transfer function Φ is defined to transit one field F_i to another one F_j, between any two scalar fields and attribute fields at the same point p:

$$F_j(p) = \Phi(F_i(p)) \tag{3.2}$$

In particular, such a definition could further define transfer functions from a scalar field to a attribute field, and vice versa.

- **Semantic Transfer Functions**

A *semantic transfer function*, $\Omega : \mathbb{E}^3 \longrightarrow \mathbb{E}^3$, is a function the defines the transformation between scalar field and attribute field at every point p and q in $\mathbb{E}^3$:

$$F_j(p) = \Omega(F_i(q)) \tag{3.3}$$

In comparison to the definition of *spatial transfer function* Ψ in [89] which operate on the same point p between the same attribute field of a spatial object, $A'(p) = A(\Psi(p))$, the above *semantic transfer function* can transfer an attribute $F_i(p)$ at a point p to a different attribute $F_j(q)$ at a different point q. Therefore, this semantic transfer function integrates the properties of conventional transfer functions Φ and spatial transfer functions Ψ.

The use of semantic transfer functions differs from that of conventional transfer functions and spatial transfer functions in two ways:

- A semantic transfer function operates on both spatial position p and field value $F_i(p)$; whereas, a transfer function can only manipulate field values on the same position, spatial transfer function can only transfer the same scalar value from one position to another.

- Manipulating volume objects becomes more flexible. Conventionally, the priority order for evaluations of a scalar field before and after spatial transfer must be considered and carefully designed, for instance, the process like normal vector calculation and spatial filtering.

As we will explain later, plenoptic functions are implemented as instances of *semantic transfer functions*, to transfer colour information at the positions on 3D plenoptic surfaces onto the colour attribute fields at 3D positions within the volume objects. In addition, a novel volume splitting function, *projective spatial functions* will be offered in the following chapters. The volume space will be split semantically, which will be of benefit to problems such as texture smearing, texture penetrating, and volume self-occlusion.

3.2 Projective Texture Models

3.2.1 Volumes and our approach

Applying textures to 3D objects has traditionally been performed in one of two distinct ways. One way is to construct a parametric map of the surface, and use the resulting *uv*-parameterization to modify the appearance by reference to a texture image. An alternative approach is to define a function which generates a texture value everywhere in space.

Our work has the following requirements for texturing volume objects: (1) Texturing volume objects with realistic appearance (projective colour map plus shading information.); (2) using a method which is robust to topological artefacts; (3) taking advantage of scalar-field based temporal and spatial constraints and transfer functions to assist texture mapping.

With the above considerations, and starting from Winter's projective texture mapping method [6], we address the issue of the boundary between image based texturing methods and solid based texture mapping approaches:
(1) We avoid full 3D procedural textures. We use plenoptic mapping functions for applying 2D texture images both to surface and to volumes within 3D datasets.
(2) Our approach avoids degradation of rendered image quality and can cope with varying texture resolution, both in the choice of source texture and in the dynamic changes which occur across the volume being rendered. The approach also allows data dependent texture mapping, through a triangular interpolation which usefully preserves edges and detail in the mapped texture. We approach this in the broader context of work on discretely sampled object representations (DSORs) [1]; that is, volume datasets and other similar forms.

3.2.2 Intermediate template

We need to place a texture on an iso-surface, ensuring it aligns with recognisable features. For consistency with the established literature, we will call any image mapped onto a surface a "texture image" and call the process "texture mapping". However, what we are really doing is mapping colouring onto the volume data, which retains its own geometric shape including any genuine three-dimensional texture.

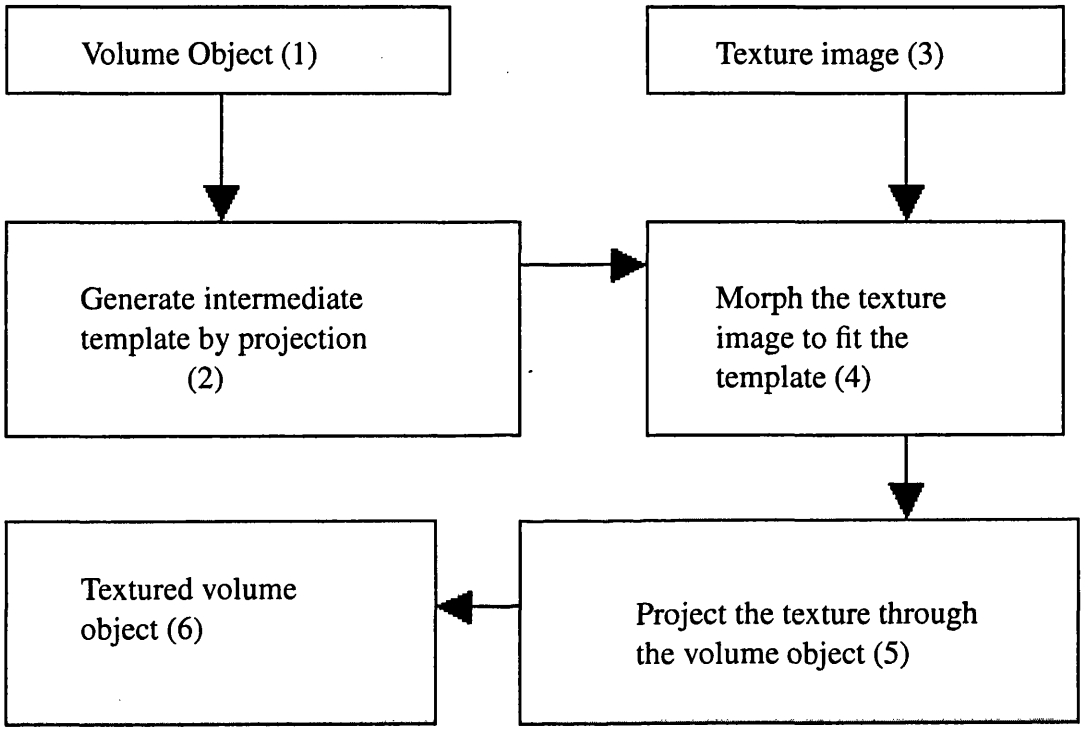


Figure 3-1: The pipeline for three-part texture mapping.

Our method does not lose or seek to hide this real texture. We first consider placing a texture image on a surface of the volume dataset. The surface has to be identified by an appropriate method for the application: for purposes of discussion, we will assume this is an iso-surface S. We also assume that we have selected an image to use as a texture. Figure 3-1 summarises the complete process, including later steps about to be described. The complete process includes the following steps:

1. Identify the surface to be textured (e.g. an iso-surface).
2. Choose the kind of projection.

3. Project the surface onto a 2D image, *the intermediate template*, choosing a resolution convenient for the next step.
4. Manually identify key points on the template with corresponding points on a texture image.
5. Warp the texture image to match the template image, bringing corresponding key points together.
6. Render the volume object.

First, the volume has to be enclosed in a conveniently regular shape, P, such as a sphere, cylinder or box. The shape of this is chosen heuristically, to match the general shape of the object being textured. In turn this shape determines a projection, spherical, cylindrical or cubical, with the centre of the projection at the centre of the shape. In effect, we will now use this as a plenoptic surface in order to determine the colouring to be applied to the iso-surface S. However we do not do this directly. We project the chosen iso-surface to a rectangular 2D image, which we call the *intermediate template* T, using a spherical, cylindrical or cubical projection as appropriate. The centre of projection is located within the volume data, which is then projected outwards onto the interior of the bounding shape P. Unwinding this shape to a plane yields the desired template T. This projection is similar to that used by Bier and Sloan [60] and others but generates the texture in a different way.

The pixel resolution of the template can be freely chosen but it is convenient to make the image large enough on screen for the next step, which is the only step requiring direct user interaction. Here, the user manually identifies key points on the template with corresponding points on the texture image. The texture image is then warped to bring the key points to the same positions as on the template image. What we have done here is use a multi constraint-based forward projection of the 3D data to get the template, then warped the texture to arrange key points on the texture image to align with the template image.

The use of key points adds a degree of elasticity to the texture layer, permitting two advantages. First, we can match the texture to the surface so that, for example, eyebrow texture sits at the appropriate position on the underlying head. Second, we can use this to cope with parts of the surface which differ greatly in slope from the plenoptic surface. When the underlying surface is roughly parallel to the plenoptic surface, the texture pixels are not stretched and a satisfactory result occurs. When

this is not the case, a single texel may be stretched across a significant area of the underlying surface, producing a very soft, smeared effect. In our approach we can adjust the texture warp to ensure sufficient texels cover the area in question, guided by the projected surface, giving a much more uniform coverage.

We now have a texture which will cover the projected shape. However, we retain the mappings rather than build that plenoptic shape directly and so we are working in a continuous space, not in a pixel-based one.

3.3 Method for Rendering

So far we have described how we manipulate the texture, in a first step. Now we address how we render the volume object so that the texture appears on it, using two further steps. Logically the remaining two parts map the texture onto the plenoptic surface and then map that onto the object. In practice rendering drives this in the inverse order. We will first assume that we wish to texture an iso-surface. Later we will address the issue of texturing the volume itself.

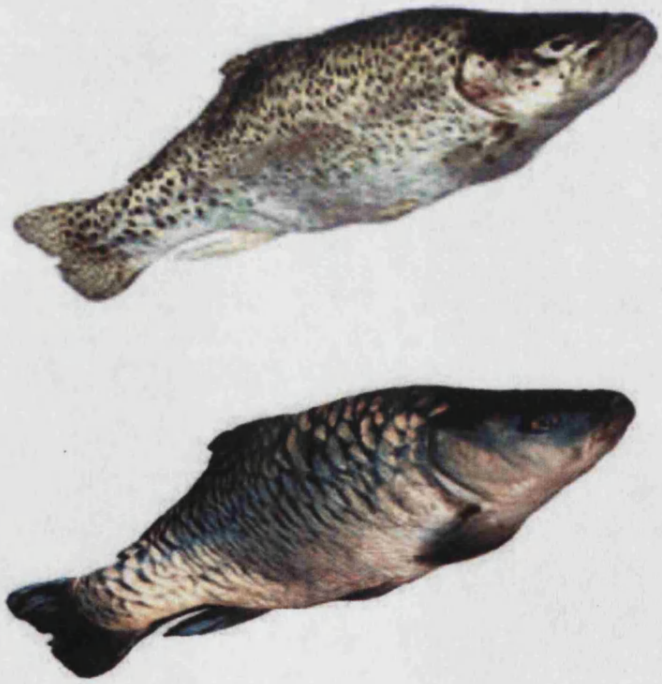


Figure 3-2: Two texture images applied to one volume object.

The second step starts by ray-casting through the pixels of the desired output image plane, to the chosen iso-surface S. At each intersection s with S we construct a direction in space which starts from the centre of the texture projection and passes through s. We follow this direction to determine where it hits the intermediate surface P (sphere, cubic, cylindrical, planar, or other implicit/explicit representations) . This yields coordinates in continuous space. The third step is to map these coordinates onto the warped texture image, again in continuous space, in order to retrieve a colour. The warped image of course consists of discrete pixels, so we interpolate those pixels which sit around the intersection, to estimate the colour at the intersection. This colour becomes the underlying colour of the surface being ray-cast. (We will say more on the interpolation process later.) The usual lighting calculations are then applied. The fish in Figure 3-2 is one volume dataset rendered twice, with different textures. In particular, one texture is not that of the real fish but is adjusted to fit using the key point method. The same approach can be used with arbitrary textures.

Strictly we are volume rendering, so voxels not on the iso-surface can also play a part in the final pixel colour. These can be made transparent, as we have done here, or treated in some other way appropriate for the application, such as accumulating the density. We give examples of partial transparency later.

3.4 2.5D Volume Textures: Pseudo-Solid Texture Model

Note that the conventional two-part texture mapping uses 2D image texture arranged to cover a 2D surface, albeit one defined in 3D-space. In comparison, solid texturing in the traditional sense is defined analytically anywhere in 3D-space, either by a 3D-space function (often called procedural solid texture) or by discrete samples in the three-dimensional space (often called 3D texture or solid texture). Here, we take a different approach, assuming that there is no such function and that we wish to use image texture. For example, there are applications where the surface texture needs to be retained to show a relationship with the original surface or to help the viewer with the geometry of the space containing the volume data. So in our implementation we define texture in 3D space by projecting a 2D texture through the volume. We refer to this as *2.5D texture*, or *pseudo-solid texture*.

Assume *p(X,Y,Z)* is a point in the volume object and the centroid position of a plenoptic primitive is (O_x, O_y, O_z), then the texture value at the position *p(X,Y,Z)* can

be indexed by a ray passing through *p(X,Y,Z)* and (O_x, O_y, O_z). This ray intersects the morphed texture image at point *S(u,v)* . The texture value (colour information, or, for instance, the perturbation coefficients for bump mapping) in the morphed texture image is then calculated and assigned to the position of the point *p(X,Y,Z)*:

$$\begin{aligned}\{R(X,Y,Z), G(X,Y,Z), B(X,Y,Z) | p(X,Y,Z) \in \mathbb{E}^3\} \\ = \{R(u,v), G(u,v), B(u,v) | S(u,v) \in T\}\end{aligned} \tag{3.4}$$

Here the intermediate template T is constructed exactly as before. The continuous coordinates (u, v) are used to locate the colour informations in the warped texture image. As shown in Equation (3.4), the rendering process calculates a direction from the centre of projection (O_x, O_y, O_z) through the ray-traced surface intersection $S(u, v)$. All other intersections lying on that radial direction will thus have the same texture colour. Note that this is a continuous space solution: we are not simply picking the nearest texture pixel but interpolating according to where the required sample lies within the morphed texture. Many samples will therefore have values not in the original pixel texture but which will be blends between those values. More importantly, as we render the volume, the radial direction will be sampled nearby rather than having samples directly along its direction. Our approach will correctly blend colours to ensure greatly reduced sampling defects.

The texture colour $\{R(u, v), G(u, v), B(u, v)\}$ can thus be extruded according to the chosen projection shape P. It can be used for non classical applications, such as in film special effects. For example, to retain the outward appearance of a face while the shape goes from normal to the underlying skull it is only necessary to use an extruded skin colouring while dynamically adjusting the selected surface.

Figure 3-3 illustrates the range of possibilities with our volume texture, using a single dataset and a single colouring set. Figures 3-3(a) and 3-3(b) are generated by a direct volume renderering while 3-3(c) and 3-3(d) use a direct surface rendering [72]. The top and bottom parts are cut off to show the projective space texture defined within the volume object. The DVR images clearly show how the texture extrudes while the DSR images leave only the chosen surface coloured.

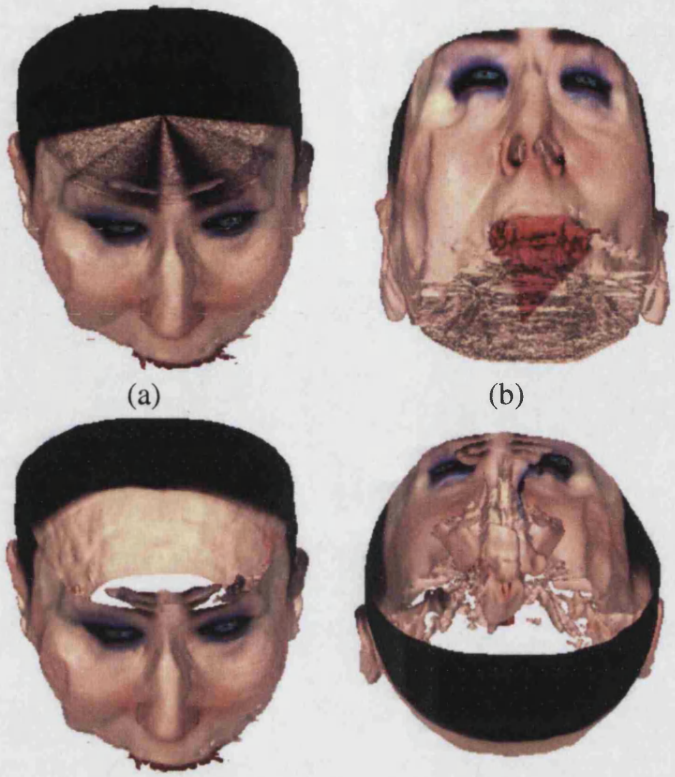


Figure 3-3: Projecting a texture image through a volume: The top and bottom parts of the volume object are cut off to show the texture projection within the volume object. Images (a) and (b) are rendered using DVR. Images (c) and (d) are rendered using DSR.

3.5 Multiple Textures on One Surface

We now extend the method to the case where we wish to apply more than one texture to a single surface. We identify regions on the template T for each texture. When rendering, it is now additionally necessary to perform an "inside" test of the template intersection against these various regions and use the result to select the appropriate texture colour. (In the case of image mattes, soft edged mattes – with blends – are needed to avoid stepped edges appearing.)

A continuous position (normalized) in the intermediate template is defined as (u,v). The discrete position of a pixel in the intermediate template is defined as (x,y). The integral coordinate pair (x,y) is indexed by u, v, where $u=x/(x_d-1)$, $v=y/(y_d-1)$, x_d and y_d

are the dimensions (the width and the height) of the intermediate template, $0 \leq x \leq x_d$, $0 \leq y \leq y_d$.

In general each texture will have a different intrinsic resolution and map to a different scale on the surface. As we use interpolation of the texture rather than choosing the nearest pixel, this is not a technical issue but it is important to both image quality and usability that this is indeed supported.

3.5.1 Multi-resolution projection of DVR and DSR

As mentioned above, the parameterization of the plenoptic surfaces could be defined by two continuous variables, $u \in [0,1]$ and $v \in [0,1]$. Therefore, it is possible to generate the intermediate template, in different resolutions, using Direct Volume Rendering (DVR) or Direct Surface Rendering (DSR) [72]. The resolution intervals of the intermediate template are defined as $\Delta u = 1/(x_d - 1)$ and $\Delta v = 1/(y_d - 1)$. Then the pixel at the position (u,v) can be indexed by the integral coordinate pair (x,y), $u = x * \Delta u$, $v = y * \Delta v$. The continuous variables u and v are used to adjust the origin and the direction of the tracing ray in the forward projection, thus different resolution intervals can be used to generate the intermediate template.

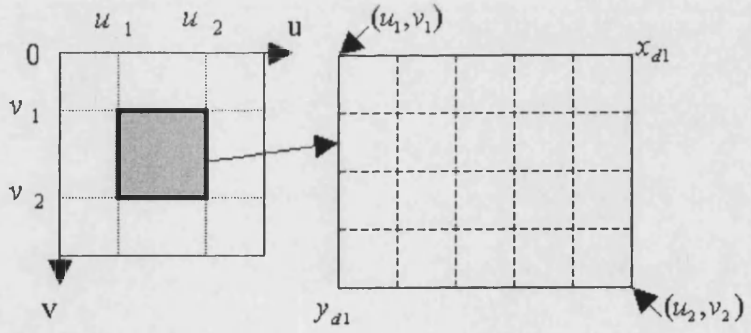


Figure 3-4: Multi-resolution Projections.

In addition, a different high-resolution representation of a rectangular area in the template image can be generated. As shown in Figure 3-4, (u_1,v_1) and (u_2,v_2) are two different positions in the intermediate template, such that $u_1 < u_2$, $v_1 < v_2$. Then, with new higher resolution intervals, $\Delta u_1 = 1/(x_{d1} - 1)$ and $\Delta v_1 = 1/(y_{d1} - 1)$, the new origin of the tracing ray is:

$$Origin(u,v) = (u_1 + x_1 * (u_2 - u_1) * \Delta u_1, v_1 + y_1 * (v_2 - v_1) * \Delta v_1) \qquad (3.5)$$

where x_{d1} and y_{d1} are the dimensions of the new high- resolution image of the region, and x_1 and y_1 are the integral coordinates, $0 \leq x \leq x_{d1}-1$, $0 \leq y \leq y_{d1}-1$.

The multi-resolution representation of intermediate template provides the possibility of high quality composition of texture images in 3D space, as demonstrated by Froumentin and Willis in their 2.5D rendering and compositing system [110].

3.5.2 High resolution patching

During rendering, textures can be assigned to positions within the volume object by using the pseudo-solid texture model, i.e., the indexing of an inverse plenoptic function. Projective texture mapping connects the 3D position $p(X, Y, Z)$ in the volume object and the 2D projected position on the plenoptic surface $s(u, v)$. Note that continuous space $P = \{p(X, Y, Z), p \in \mathbb{E}^3\}$ by definition can have infinite texture details. Therefore the higher resolution texture image could be continuously projected to the projective space without any limitation. This mechanism we call *high-resolution patching*. Details of conventional solid textures are limited by their grid resolutions.

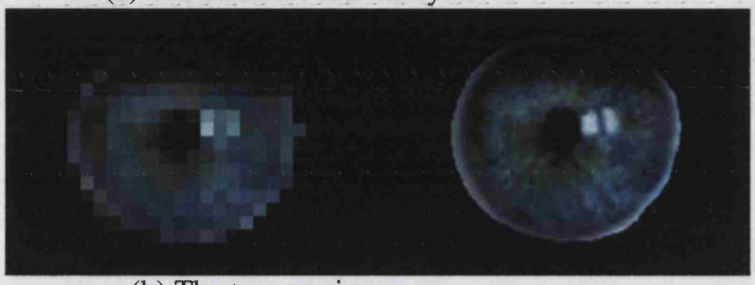
We assume the high-resolution patch is positioned at a rectangular area in the morphed texture image, which is represented by the coordinates of its two vertices, (u_{a1}, v_{a1}) and (u_{a2}, v_{a2}), where $u_{a1} < u_{a2}$, $v_{a1} < v_{a2}$. Then the position, (x_a, y_a), in the high resolution patch, can be defined as:

$$(x_a, y_a) = ((u - u_{a1})/\triangle u_a, (v - v_{a1})/\triangle v_a) \tag{3.6}$$

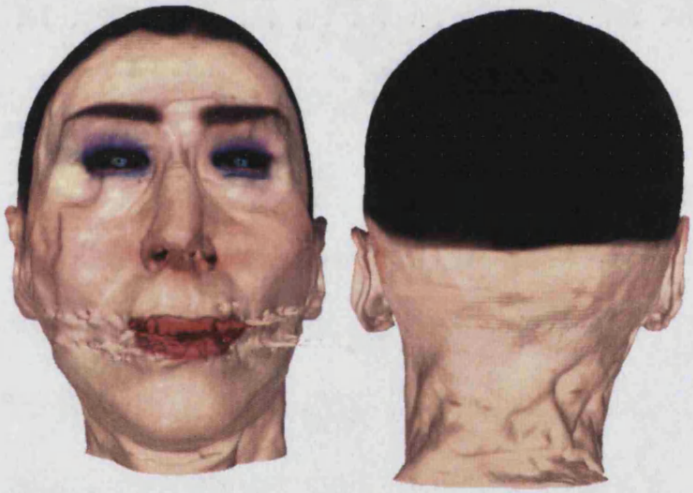
where $\triangle u_a$ and $\triangle v_a$ are the resolution intervals of the high resolution patch, which are represented by the width and the height of the patch. *(u,v)* are the coordinates of the position in the morphed texture image.

The above two equations, Equations (3.5) and (3.6), are not just the resolution representations of an image. Equation (3.6) is used to construct pseudo-solid textures during volume rendering. Equation (3.5) is used to adjust the origins of tracing rays in DVR or DSR [72], on plenoptic surfaces. These two equations are different and independent texture indexing functions. They are implemented in the processes of inverse projection and forward projection respectively.

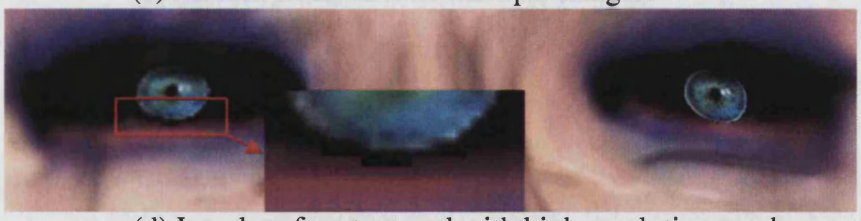
(a) The areas around the eyes.



(b) The two eye images.



(c) Surface textured with multiple images.



(d) Local surface textured with high resolution patch.

Figure 3-5: Use of higher detail texture in critical parts of the image. In (d) the left image of the eye has sampling defects not present in the right, high resolution image.

In Figure 3-5 we show how this makes a difference to the eyes. Even though the difference cannot be seen in longer shots, it becomes apparent as we zoom in. Figure 3-5(a) is a rendered image of the area around of eyes. We use this image as the

intermediate template to locate the position of the rectangle areas for texture patching. Figure 3-5(b) shows two images of the eyes. The left image is in low resolution (30x21 pixels), the right image in high resolution (800x564 pixels). Figure 3-5(c) is a rendered image of the volume head. Figure 3-5(d) shows the close up of the textured eyes. The texture of the left eye (pseudo-solid textures constructed through low resolution patching) loses details and suffers from pixelation (as shown in the area enclosed in the red frame). However, the texture of the right eye (pseudo-solid textures constructed through high resolution patching) preserves all the detail and shows no such defects.

3.6 Different Textures on Different Iso-surfaces

Extending the method to having different (groups of) textures assigned to different selected iso-surfaces is now straightforward: we simply need a set of textures for each iso-surface to be textured. Figure 3-6 shows examples, each using pairs of textures. The beauty of this consolidated texture mapping method is the flexibility provided by combining a variety of novel semantic constraints, not just the benefits provided by spatial constraints demonstrated by Dr Winter in his PhD thesis [6].

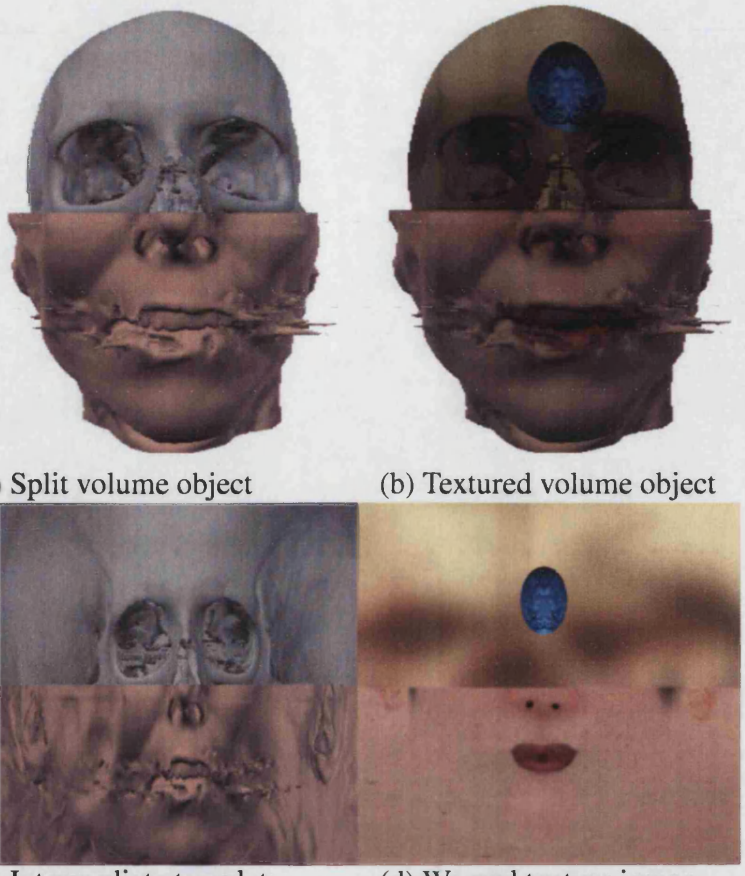
When using our approach with direct volume or surface rendering, semantic constraints can be used to facilitate the texture mapping process. We demonstrate this with a spatial constraint to split the iso-surface into different portions. Each portion is then associated with its own texture. In Figure 3-6(a), the skin of the head has two texture images. The top and the bottom parts of the head are split [6, 14]. Then, if the 3D position on the tracing-ray locates within the space of the top half, this 3D position is indexed onto one texture image. Similarly, if the 3D position locates within the space of the bottom half, it is indexed onto another texture image. Figure 3-6(b) was rendered in the same way, using different spatial constraints and different texture images.

(a) (b)

Figure 3-6: Multiple surfaces, separately mapped. (a) The top half and the bottom half of the head are textured by the use of two different spherical texture images. (b) The surface of the earth is textured by a spherical texture image and rendered using CVG operations. The core of the earth is textured by a cylindrical texture image. The transparency of the air is controlled using a field function.

In order to generate the warped texture images for each iso-surface, we need in principle to generate an intermediate template for each of them. However, since the mapping is implemented via a volume rendering engine, and a constraint is used to split the volume object, one intermediate template may include the key points of multiple iso-surfaces and we need only one texture image to texture multiple iso-surfaces. In other words, the intermediate template reflects the constraints placed on the renderer.

This is illustrated in Figure 3-7(a), where the volume object has two iso-surfaces, the skin and the skull. The top half of the skin is removed, using a spatial constraint, as shown. Figure 3-7(b) shows the textured output. In order to produce this, we generated an intermediate template from the constrained data. This template is shown in Figure 3-7(c). Then we produced a warped texture in the usual way, Figure 3-7(d), for the final rendering.



(a) Split volume object (b) Textured volume object

(c) Intermediate template (d) Warped texture image

Figure 3-7: Spatial and semantic constraints within one intermediate template.

3.7 Texture Interpolation

This is an appropriate point to discuss texture interpolation further. The issue is, given a texture image defined at discrete positions (pixels), interpolate the value at any position in between.

Interpolation methods permit texture to be sampled anywhere at a point in a continuous plane, or supersampled at multiple continuously-positioned points, even though the source image is discrete. They thus sit comfortably between discrete texture and procedural texture while still being image-based.

What interpolated methods have in common is a combination of a (discrete) pixel image and a (continuous) interpolation function. The pixel image determines the resolution limit but the choice of function determines the image quality achieved. Since

all such methods start with a pixel image, the choice of function also characterises the specific method.

Bilinear interpolation is well-established and supported in graphics cards, where it is used to texture the surfaces of scenic objects in games, for example. Image quality is less important in fast-moving games than in visualisation, so some softening of the texture is acceptable.

An alternative approach [66, 67] explores a method based on interpolating only three of these pixels. The motivation is better reconstruction of edges, retaining perceived sharpness and fine detail. Briefly, the method pre-processes the texture image to add one bit per pixel. Each group of four pixels is examined to determine the pixel with the outlier brightness.

A notional diagonal is then placed across the four pixel square, such that the outlier is strictly to one side of the diagonal. In other words, the diagonal is the one which does *not* include the outlier. This diagonal roughly represents the direction of any edge in that square.

As there are only two possible diagonals in each four pixel group, one bit suffices to identify which. To sample the texture, the triangle that the continuous space sample falls in is determined. The three pixel values are interpolated to determine the value at the continuous point. This interpolation is the same method that Gouraud shading uses. A modest extension to the pre-processing changes the direction of a diagonal if its neighbours are majority in favour of the other direction. This captures visible edges better and makes no difference to the storage requirement or to the rendering method or speed. The approach has also been demonstrated in matted (layered) images [110, 111].

We adopt this way of interpolating texture images because it readily adapts to being sampled at varying resolution (required because the template is not a uniform-area mapping of the iso-surface) and because visualisation applications are more demanding of visual quality than computer games. As a bonus, it reduces visual sampling defects and is very fast. The method also performs well dynamically, with no discernible flicker while zooming [66].

3.7.1 Pixel level data dependent interpolation

As shown in Figure 3-1, the warping and morphing operations implemented in steps (3) and (4) may degrade the quality of the 2D plenoptic projected texture image, especially the quality of the tiny texture features in the image. Therefore pixel level data-dependent interpolation [66, 67] is adapted in the warping and morphing operations in our algorithms, to improve the quality of the morphed texture image.

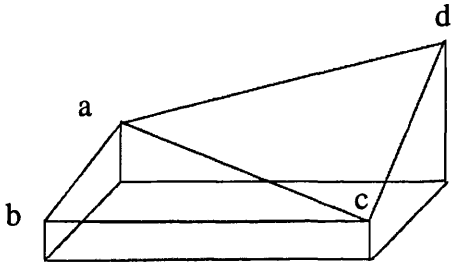
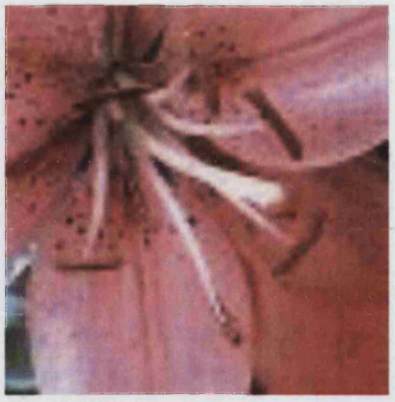


Figure 3-8: Pixel level data dependent interpolation: *a, b, c, d* are four pixel values, represented as heights [66].

Assume Figure 3-8 shows four pixel values, represented as heights. Suppose the (low resolution) plenoptic projected image (or a texture photo directly) is *X*, the high-resolution target image to be generated is *Y*.

First, scan *X* to initialize a 2D array, x_{mn}, which records the edge direction of all four-pixel squares. Second, scan *Y* to initialize a 2D array, y_{ij}. Each y_{ij} is inversely mapped to the sample image *X*, and the array x_{mn} is used to identify the triangle in which the point falls in the source image *X*.

Then the value of y_{ij} is calculated by the use of barycentric interpolation. In the first step, the outlier is detected by comparing the difference $|a - c|$ and the difference $|b - d|$. The pair with the smaller difference is treated as an edge. This maintains the smoothness within the regions as well as the sharpness between the flat region and cliff region. Pixel level data dependent interpolation preserves edge details by keeping edges sharp and smooth areas smooth. Figures 3-9 and 3-10 give an example of the application of the interpolation method. For comparison purposes, the bilinear interpolation is also implemented in our algorithms.



(a) Stamens

(b) Textured volume object.

Figure 3-9: Pixel level data dependent interpolation: (a) The original photo of stamens, 75 x 75 pixels. (b) Textured volume object. The stamens on the left side are interpolated by the pixel level data dependent interpolation, the stamens on the right side by bilinear interpolation.

Figure 3-9(a) is the original photo of stamens. The image size is 75x75 pixels. Figure 3-9(b) is the textured volume object. The stamens on the left side in Figure 3-9(b) and Figure 3-10 are interpolated by the pixel level data dependent interpolation, the stamens on the right side by bilinear interpolation. The close up view given in Figure 3-10 shows that the pixel level data dependent interpolation gives the better appearance and details of the textures are preserved quite well. Figure 3-10 shows the enlarged image of the textured object, which is shown in Figure 3-9(b).

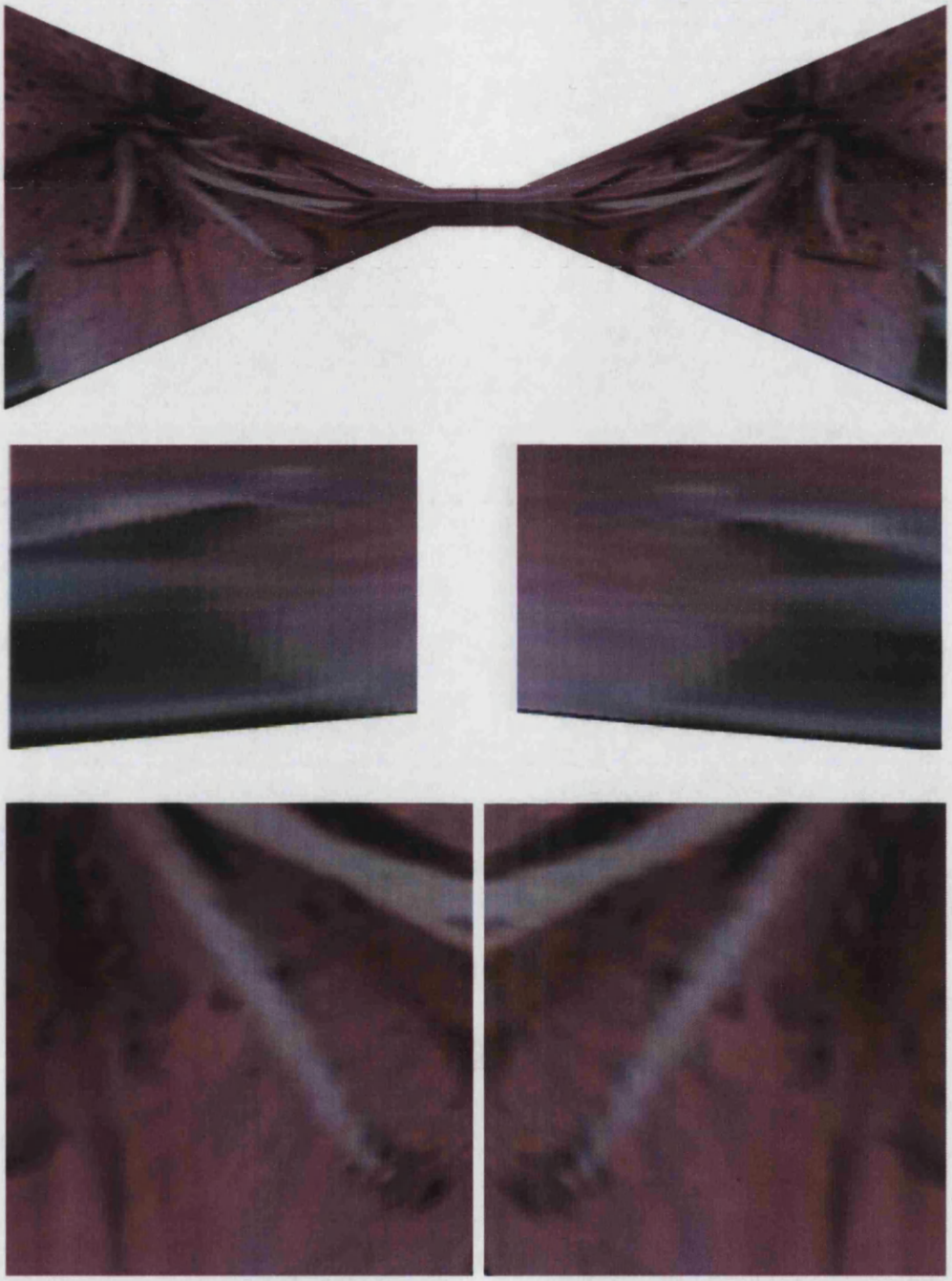


Figure 3-10: The rendered image of textured volume object. Left stamens: pixel level data dependent interpolation; Right: bilinear interpolation. The deformation of the volume object is through spatial transfer function. The volume object is shown in Figure 3-9(b).

3.8 Further Experimental Results

We show here some more results, combining various aspects of the complete system.

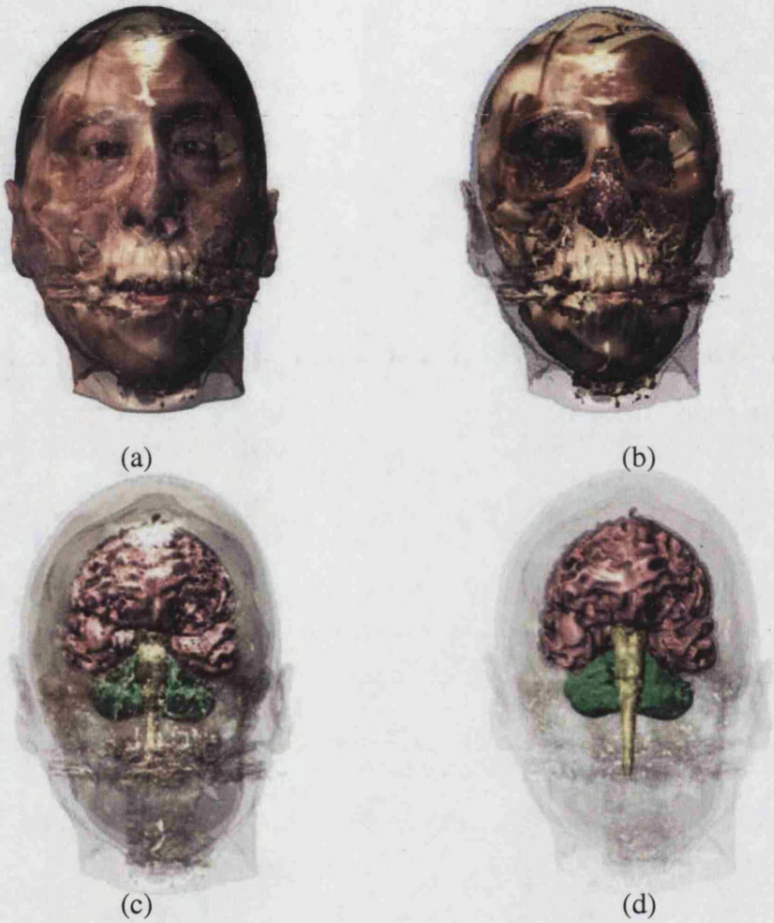


Figure 3-11: Multiple iso-surfaces associated with their own projected textures.

3.8.1 Multiple textures, varying opacity

The core ideas are brought together in Figure 3-11. These images are taken from a short movie sequence, in which the composite volume object is rendered with gradually varying opacities of the iso-surfaces. Each iso-surface is associated with its own texture set. In Figure 3-11(a), the opacity of the skin is 0.8, the opacity of the skull is 1.0. In Figure 3-11(b), the opacity of the skin 0.2, the opacity of the skull 0.9. In Fig-

ure 3-11(c), the opacity of the skin is 0.15, the opacity of the skull is 0.3, the opacity of the brain is 1.0. In Figure 3-11(d), the opacity of the skin is 0.05, the opacity of the skull is 0.1, the opacity of the brain is 1.0. The skin is coloured using a cylindrically-projected photograph of a person, the skull is coloured using a cylindrical mapping of the image of a real skull. In Figure 3-11(c) and (d), the brain has three volumes: the main volume, the vertical stem, and the rear lobes. The colours of these are directly assigned to density ranges.

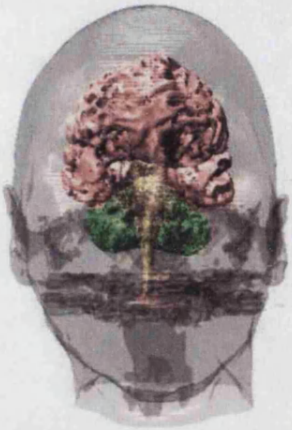
Here there is an outer surface of skin. Within that there is a skull surface and inside that a brain surface. These surfaces are determined solely by the volume data but our choice of colouring determines how they are rendered. It is important to remind ourselves that the real texture of the surfaces is still present, from the volume data. Only the colouring is "texture" in the sense that the computer graphics community uses the term. This example shows that the transparency can be adjusted gradually, to reveal the coloured surfaces one by one. It also illustrates how our key point selection method can be used to adjust the texture images to match the volume data. For example, the face is not that of the cadaver from which the volume data was captured but has been adjusted to fit the volume data. This is again relevant to special effects, where the face will be that of an actor but the skull and other internal detail can be obtained from other sources or indeed synthesised as something wholly unrelated, such as circuitry to represent the interior of a robot.

3.8.2 Spatial constraints and transfer functions

Rendering is now slightly more complex because we need to decide which surfaces are visible and whether they are opaque or translucent. In addition, radial projections of textures might be inappropriate in some applications. Note that, in our system the semantic-constraints based projective texture mapping mechanism is implemented through volume rendering methods (DSR [72], DVR [112]), so spatial constraints and transfer functions facilitate and control the positions of the texture. These are all within the grasp of a volume renderer.

Figure 3-12 shows how multiple iso-surfaces can be associated with their own texture images, opacity settings and spatial constraints.

(a) Textured skin



(b) Semi-transparent skin plus coloured brain

(c) Textured skull

(d) Semi-transparent skull plus coloured brain

(e) Rendering with spatial constraints -1 (f) spatial constraints -2

Figure 3-12: (a)–(d) Multiple textures applied to the skin, the skull and the brain. In (e) and (f) different portions of the skin and the skull were set to transparent using spatial constraints.

In Figure 3-11, the semantic constraints of the iso-surfaces were achieved by using field functions. In Figure 3-12, the semantic constraints of the iso-surfaces were achieved by using spatial transfer functions. Similarly, as shown in Figures 3-5, 3-6, 3-7, and 3-14, spatial constraints (3D positions, splitting functions, etc.) were employed while texturing and rendering volume objects. Volume data offers us direct control of texturing from the data itself, whereas in surface methods external geometric constraints have to be designed.

3.8.3 Sculpting with 2.5D texture

Animators are used to seeing the external appearance of an object being animated, even when sculpting to help design the shape. Using our 2.5D method the texture can be applied and the shape then refined by sculpting, while still showing the correct texture in the newly carved regions. If any minor repositioning of the texture is then needed, it can be made on the final shape. This is much easier for the animator than sculpting a blank form. Figure 3-13 illustrates a tapered cylinder (a volume object) textured this way. The shape of the cylinder has been modified by sculpting, with the texture appearing much as it would for carved solid texture. This texturing operates the same as that for sculpting objects with procedural solid texture, except that procedural textures do not suit this kind application. In contrast, our 2.5D approach is directly appropriate and compatible with the existing way of texturing the head.

Figure 3-13: Sculpting textured volume object: the 2.5D texture model provides pseudo-solid texture for volume objects.

3.8.4 Semantic (spatial, logic) constraints for 2.5D texture models

Spatial Splitting

Projecting 2D texture to 3D voxels is an ill-conditioned problem. The texture may wrongly be applied to parts of the volume where it is not needed. Since the presented three-part texture mapping method can be directly hooked into volume rendering algorithms (DSR and DVR), we can make use of spatial transfer constraints or semantic constraints [14]. We use splitting techniques to isolate the portion of the volume object which needs to be textured (as demonstrated in several of the previous figures). Now we consider the case that once the volume object is textured using our method, splitting techniques can be directly used for the visualisation. Figure 3-14 demonstrates the effect of splitting the textured volume object. The outer layer was split using semantic constraints. As can be seen in Figure 3-14, the textures extrude through the split layer as well as the volume object itself. The renderer can thus assist us to choose how the extruded texture is applied, according to the application area.

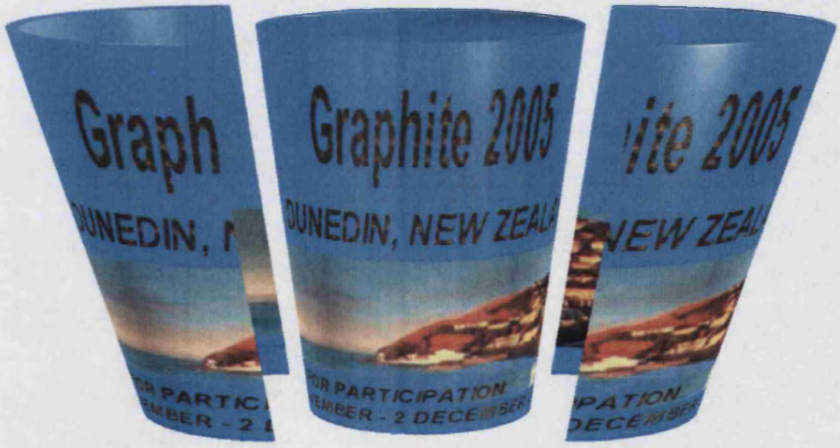
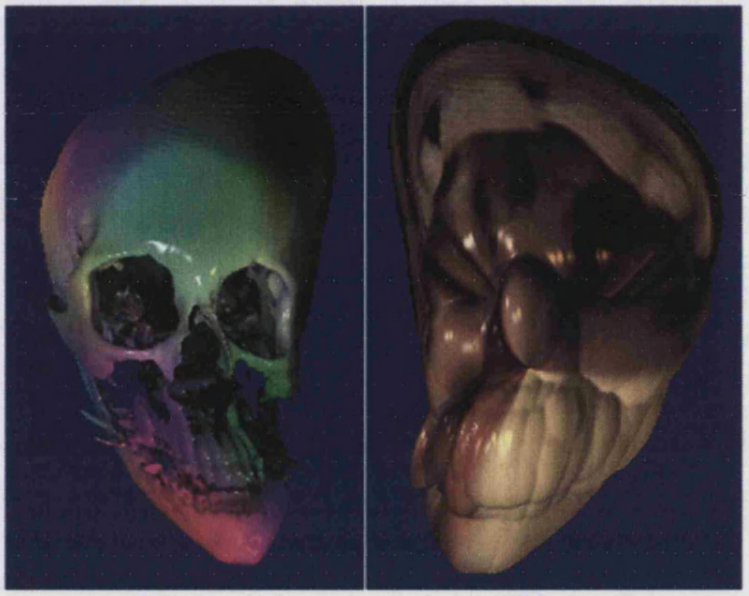


Figure 3-14: Common 2.5D texture on semantically-split layers.

Logic and field functions

Note that one benefit of the generic volume model is flexibly manipulating its components through transformation functions. In its definition, components can be a variety of segmented volume elements. Transformations can be a general set of transfer functions, spatial functions, deformation functions, etc. Here, we use a distance field based skull model to demonstrate the flexibility and generality of semantic constraints. As shown in Figure 3-15, a distance-field based skull is rendered using a procedural

texture model, a cylindrical texture model, logical (distance value) constraints, a field function, and FFD deformations.



(a) (b)

Figure 3-15: Coloured skull: Distance-field based skull can be logically, spatially, and semantically split. Different texture model such as 2D images (right), procedure textures (left) are applied to tagged volume objects accordingly. The skull model is deformed using FFD.

First, given a volume object, the distance field could be one of its scalar fields. The field value at position p, could be the distance from p to its nearest point on an iso-surface [17]. Thus, different distance values in fact provide us with useful depth information for splitting the volume space.

Second, we combine FFD deformation control [89] with the semantic projective texture model. Free-Form Deformation (FFD) is a well-established deformation technique well-adapted to volume graphics [6, 89].

Adopting the Bézier deformation model, the volume space was discretised by imposing a 3D rectilinear grid onto it. Each vertex in the grid, where parametric coordinates, v_u, v_v, v_w, are known, is transformed into Euclidean space using the Bézier volume equation:

$$P_B(v_u, v_v, v_w) = \sum_{k=0}^{3}\sum_{j=1}^{3}\sum_{i=0}^{3} C_{i,j,k} B_i(v_v) B_j(v_m) B_k(v_m) \tag{3.7}$$

where $(v_u, v_v, v_w) \in [0\ 1]^3$ is the parametric coordinate of the vertex, $C_{i,j,k}$ is a Bezier volume control point and B_i, B_j, B_k are the Bernstein functions.

Each vertex in the grid maintains a record of both the original parametric (v_u, v_v, v_w) and the computed Euclidean coordinates (v_x, v_y, v_z). Hence each cell encloses a small parametric domain, which is further divided into six tetrahedra. Spatial transfer functions can be generated within such parametric space using bary-centric interpolation. The problem of determining (v_u, v_v, v_w) is thus transformed to the search for a tetrahedron containing v.

Third, the procedural texture model was defined as follows:

$$R(p_u, p_v, p_w) = (vluNoise(p_u * 5.0, p_v * 5.0, p_z * 5.0) + 1.0)/2.0 \tag{3.8}$$

$$G(p_u, p_v, p_w) = (vluNoise(p_v * 5.0, p_u * 5.0, p_z * 5.0) + 1.0)/2.0 \tag{3.9}$$

$$B(p_u, p_v, p_w) = (vluNoise(p_w * 5.0, p_v * 5.0, p_u * 5.0) + 1.0)/2.0 \tag{3.10}$$

where $vluNoise$ is a solid texturing basis function provided by *VLIB*. $(p_u, p_v, p_w) \in [0\ 1]^3$ are the coordinates of a 3D point p.

In the VLIB implementation, the noise basis function $vluNoise$ is implemented through the summation of pseudo-random spline knots. The knots were calculated using loop calculation and a lookup table. Given a lookup table $valueTab$, the integral coordinates of x, y, z are:

$$ix = FLOOR(x) \tag{3.11}$$

$$iy = FLOOR(y) \tag{3.12}$$

$$iz = FLOOR(Z) \tag{3.13}$$

The fractional remains are:

$$frac[0] = x - ix \tag{3.14}$$

$$frac[1] = y - iy \tag{3.15}$$

$$frac[2] = z - iz \tag{3.16}$$

Then the noise value was calculated as:

$$
\begin{aligned}
&for(k = -1; k <= 2; k++)\{ \\
&\quad for(j = -1; j <= 2; j++)\{ \\
&\qquad for(i = -1; i <= 2; i++) \\
&\qquad xknots[i+1] = valueTab[INDEX(ix+i, iy+j, iz+k)]; \\
&\qquad yknots[j+1] = vluSpline(frac[0], 4, xknots); \\
&\quad \} \\
&\quad zknots[k+1] = vluSpline(frac[1], 4, yknots); \\
&\} \\
&\qquad\qquad noise = vluSpline(frac[2], 4, zknots);
\end{aligned}
\tag{3.17}
$$

where *vluSpine* is the implementation of the spline function [**?**].

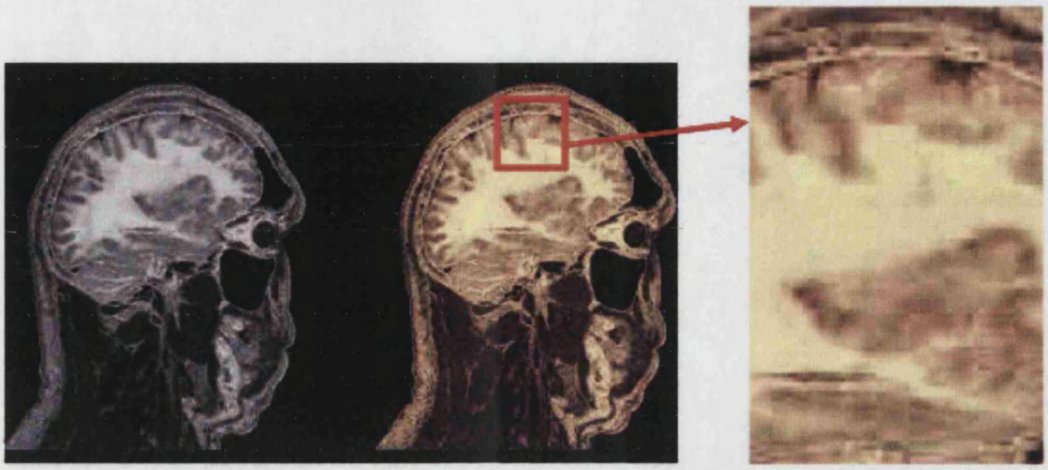
3.9 Image based Illustrative Colour Transferring

As previously discussed, a semantic transfer function can transfer an attribute field $F_i(p)$ to a different attribute field $F_j(q)$ at different positions. Positions p and q can be either 2D or 3D positions. Transferring colours from an illustrative image to 2D slices of volume objects, or from 3D illustrative volume to 3D volume objects, are common tasks to annotate volume objects. These techniques are in fact different implementations of semantic transfer functions.

The common colour transfer techniques include statistical and colour correction between source and target images [113, 114] and cluster similarity based colour transfers between illustrative images and volume sources [115]. Here, the implementation of the colour transfer functions is based on cluster similarity and statistic colour corrections.

We define the source (volume) intensity (or other attribute) clusters $G_s = \{G_{si}, i = 1, ..., N_s\}$, and example (illustrative image) colour clusters $G_e = \{G_{ei}, i = 1, ..., N_e\}$. We follow Lu's assumption [115]: "Since illustration usually employs different colours for different objects, we assume that if two objects do not have the same colours in the example, they will not share the same colours in the source.", hence we assume the volume intensity cluster number N_s is equal to that of the illustrative colour clusters,

N_e.



(a) (b) (c)

Figure 3-16: Illustrative Sample Image [46]: (a) A 2D slice of the Visible Man. (b) Realistic Image of original bones, skins, soft tissues. (c) The sample area cut off from (b) (the enclosed area within red box) is used for constructing colour clusters and transferring colours from pixels to voxels.

We need to find a correspondence map f from G_s to G_e using the distribution similarity criterion between the source cluster G_{si} and the example cluster G_{ej}. Assuming clusters are interactively provided by users through the swatch technique [113], then the mapping f can be defined by finding the minimum errors between the normalised areas of source clusters and example clusters, of course, users can directly construct f as a lookup table, as illustrators usually do.

Given normalised areas $S = \{si = Area(G_{si})\}$ and $E = \{ei = Area(E_{si})\}$, the mapping function $f : f(S) \rightarrow E$ is defined as:

$$M_{ERROR} = \sum_{i=1}^{N_s} (s_i - f(s_i))^2 \tag{3.18}$$

With equation (3.18), source cluster G_{si} will be matched to example cluster G_{ej} by $j = f(i)$.

After constructing the mapping function, we can transfer colours from example clusters to their matching source clusters. We perform the colour transfer in $\iota\alpha\beta$ channels. The colour transfer can be described as follows:

(1) Convert source intensities and example colours (RGB) into $\iota\alpha\beta$ channels, ι_v, α_v,β_v, ι_p, α_p,β_p, where v is a voxel in the source, p is a pixel in the example.
(2) Calculate the mean and standard deviation of each cluster in the three converted spaces, $m_{w,Gsi}$, $std_{w,Gsi}$, $m_{w,Gej}$, $std_{w,Gej}$, $w = \iota, \alpha, \beta$
(3) Calculate the distance $d_{v,i}$ between voxel v and clusters $G_{\iota,si}$ in ι space: $d_{v,i} = |\iota_v - m_{\iota,G_{si}}|$.
(4) Calculate average coefficients $c_{v,i} = Inv(d_{v,i})/\sum_{j=1}^{N_s}(Inv(d_{v,j}))$, where $Inv()$ is the inverse proportional function of distances (contribution factors).
(5) Scale and transit $\iota\alpha\beta$ channels of voxel v:

$$\hat{\iota}_v = \sum_{i=1}^{N_s}(c_{v,i} * ((\iota_v - m_{\iota,Gsi}) * std_{\iota,Gej}/std_{\iota,Gsi} + m_{\iota,Gsi}))$$

$$\hat{\alpha}_v = \sum_{i=1}^{N_s}(c_{v,i} * ((\alpha_v - m_{\alpha,Gsi}) * std_{\alpha,Gej}/std_{\alpha,Gsi} + m_{\alpha,Gsi}))$$

$$\hat{\beta}_v = \sum_{i=1}^{N_s}(c_{v,i} * ((\beta_v - m_{\beta,Gsi}) * std_{\beta,Gej}/std_{\beta,Gsi} + m_{\beta,Gsi})) \quad (3.19)$$

where $j = f(i)$. Equation (3.19) transfers colours from example image to source volume in $\iota\alpha\beta$ space. Then we can further convert $\hat{\iota}, \hat{\alpha}, \hat{\beta}$ into RGB space [114, 116] (Ref. Appendix for the original Matlab code provided by [116]):

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix} * \begin{pmatrix} \hat{\iota} \\ \hat{\alpha} \\ \hat{\beta} \end{pmatrix}$$

In Figure 3-17, we first scale the original 16-bit intensity values of the MRI brain into 8-bit non-negative integers. We then use swatches to construct two source (intensity) clusters from a slice of volume dataset and two example (colour) clusters from the illustrative image shown in Figure 3-16 [46]. The colour components are normalised for the purpose of transferring colour space. The R, G, B colour components will be reshaped back into 8-bit integers finally. These colour components and their indexed original 16-bit intensities construct the colour lookup table. The volume rendering engine uses this lookup table to transfer colours to each voxel's red, green, and blue scalar fields.

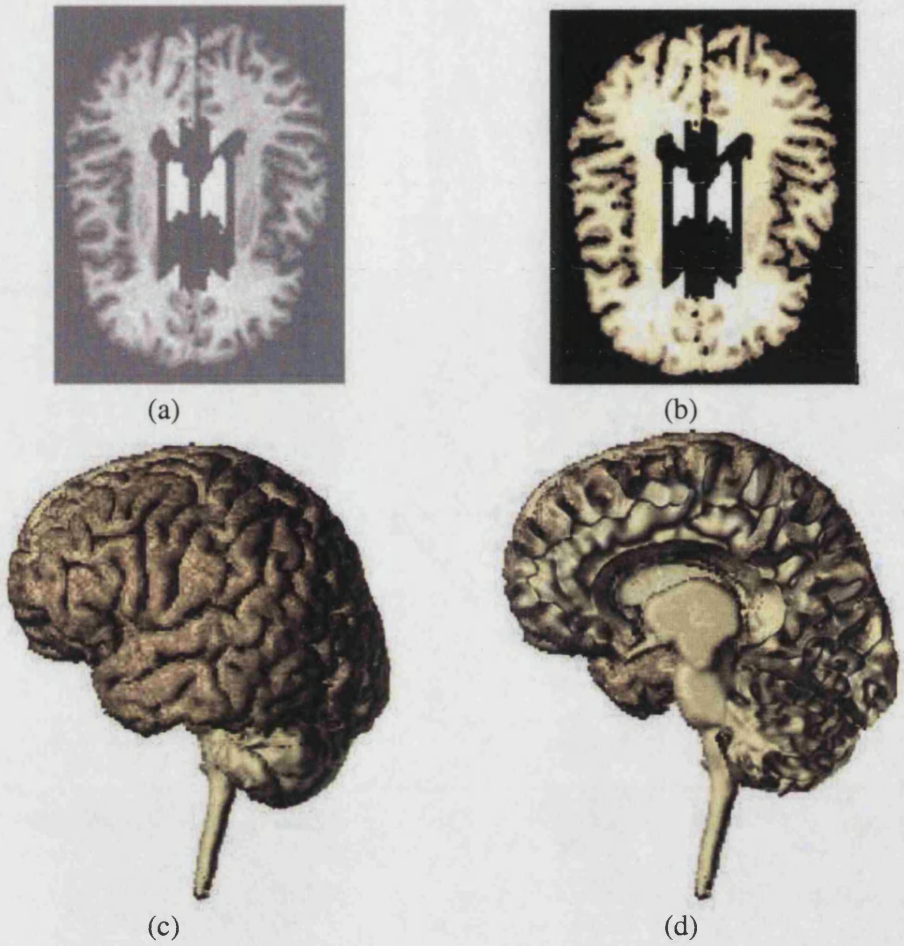
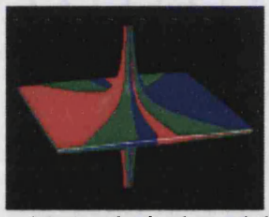


Figure 3-17: Illustrative colour transferring: (a) A 2D slice of volume MRI brain. The grey image (voxel intensities) is enhanced using histogram stretching. (b) The intensities shown in (a) were transferred to the colour information shown in Figure 3-16(c), illustrative image. (c) Rendered volume MRI brain. (d) Regions of MRI brain were cut off to show the internal structures. Colour transferring functions in both (c) and (d) are constructed using the same illustrative image.

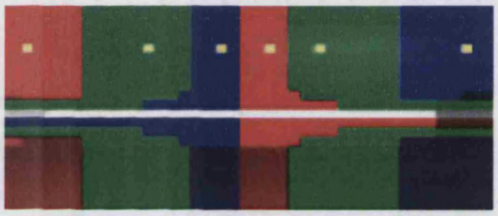
3.10 A Generic Problem: Penetrating and Self-occlusion

3.10.1 Shear-effect

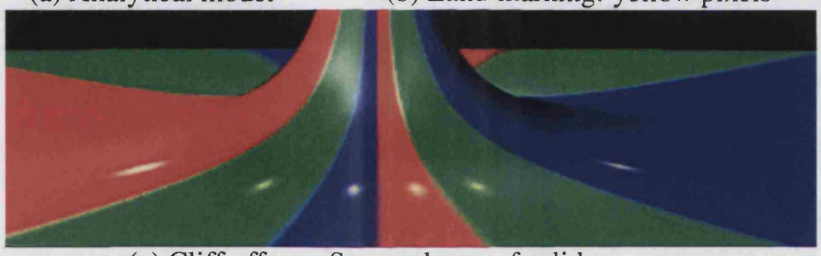
An analytical model was built to test the quality of the projective texture mapping. A typical artefact is the shear-effect, that is, a single texcel projected to a surface which is almost parallel to the direction of projection will be smeared over a relatively large area of solid. In addition, the finite resolution of colour lookup tables in procedural texture models can also introduce alias artifacts due to "texture minification" [47]. Therefore, effective surface representations as well as appropriate interpolation techniques are primary requirements of the novel semantic project texture model discussed in this chapter.



(a) Analytical model



(b) Land-marking: yellow pixels



(c) Cliff-effects: Smeared area of solid.

Figure 3-18: Shear effect of pseudo-solid texture model: (a) The constructed analytical model. (b) The morphed texture image. Land-marking dots are made in yellow. The size of each marking dot is only a single pixel. (c) is the close up views of the textured analytical object. The size of the morphed texture image used in figure (c) is 128x128 pixels.

Figure 3-18(c) shows that the shear effect becomes prominent. As previously discussed, continuous representations of texture image could be a solution to the problem of texture smearing. Instead, a more advanced and practical technique to solve the shear effect, smoothing point clouds, will be offered in the next chapter.

3.10.2 Penetration and self-Occlusion

As shown in Figure 3-3, texture penetration is a critical issue in pseudo-solid texture models. Another challenge is the *self-occlusion* of the volume dataset, that is, iso-surfaces always have manifold geometrical structures. As previously discussed, DSOR objects lack geometrical, topological and semantic information. Therefore expecting users to have expert knowledge of transfer functions to split manifold structures is unrealistic. Regarding this, we will offer a novel solution, *semantic layers*, which is based on the concept of semantic transfer functions, to solve this problem in a following chapter.

3.10.3 Computational Expenses

The volume rendering pipeline presented in this chapter is implemented using c/c++. The program is executed on the unix platform without additional parallel computing resources. The field-based illumination object model, which is based on the single scattering model, is used in the volume integral operation. The field illumination model casts a second ray (shadow ray) to calculate the shadow effects for each of the light source in the scene. We use DSR [72] and DVR [78] as our main volume rendering techniques.

The complexities of field functions, spatial functions, deformation functions implemented as cascading functions for each volume integral elements of tracing rays, will direct affect the rendering time. Rendering Figure 3-2 will take 1 minutes and 57 seconds. The image size is 400x400 pixels and the running step length is set to 0.1. There are 8 point lights in the rendering scene. The size of the carp dataset is 256x256x512 voxels.

Rendering Figure 3-5(c) will take 7 minutes and 1 second. The image size is 1320x2000 pixels and the running step length is set to 0.1. There are 4 point lights in the rendering scene. The size of the CT-head dataset is 180x113x237 voxels.

Rendering Figure 3-12(d) will take 90 minutes. The image size is 2000x2000 pixels and the running step length is set to 0.2. There are 2 point lights in the rendering scene. This image is used as an example printed on the back cover of the proceeding of Graphite2005. The size of the MRI-brain dataset is 109x189x172 voxels. The size of the CT-head dataset is 180x237x113 voxels.

Rendering Figure 3-15(a) will take 40 minutes, and Figure 3-15(b) 19 minutes.

The image size is 400x400 pixels and the running step length is set to 0.1, and there are 3 point lights in the rendering scene. The size of the distance field based dataset is 109x189x172 voxels.

3.11 Conclusion

We have considered the needs of volume rendering when being used for applications that needs realistic appearances of volume objects. We focus on volume iso-surfaces rather than mesh surfaces. We have presented a multi-constraints-based approach to texturing which is based on continuous space mappings to ensure good image quality. Starting from Winter's projective texture mapping method [6], the system requires only one intervention by the user, to determine key points where the texture must match an intermediate image of the original data. This can also be used to avoid the problem of texture being smeared over too large an area.

We demonstrated an extension to 2.5D textures, extruded through the volume, using an approach consistent with 2D texture. In conjunction with its intrinsic ability to generate high resolution images, the overall method has potential in non-classical areas, such as film special effects, for which volumetric source data are especially useful, whether captured or synthesised.

If topological artefacts exist, our methods will not degrade the visual effect of the textured volume objects. Any such regions (whether sculpted or damaged portions of volume objects) will have the same texture as their radial neighbourhood, as demonstrated in Figures 3-13 and 3-14. In contrast, as investigated by Wood [32] and Guskov and Wood [29], traditional mesh-based texture mapping needs to fill the holes introduced in the mesh model, especially for high quality texture mappings, for example in close-ups and realistic volume characters.

Interpolation techniques, continuous indexing, and multi-resolution representations, are typical computer graphics methods. We migrate these techniques from the traditional CG area into Volume Graphics. In our textured volume rendering, the iso-surfaces (point clouds) are used as semantic constraints, i.e, spatial constraints and transfer functions [14, 63, 62], to facilitate and guide the texture placement with high accuracy, giving a general purpose solution.

Being able to sculpt a volume object with texture running through it is important. Manipulating the details of volume characters is essential. When using volume source

data, removing the imaging noise and correcting topological artefacts without touching the textured realistic appearance is a challenge. Our 2.5D projective texture could possibly be hooked into practical volumetric sculpting, for example as presented by Ferley et al. [117] for modelling volume objects, or applied to volume objects whose surfaces were smoothed using diffusion normals, as presented by Tasdizen et al. [118].

Transferring colours from illustrative images to volumes also demonstrates the flexibility and generality of the described generic volume object model, which combines together the generality of scalar fields and the flexibility of a variety of semantic transfer functions.

Pessimistically speaking, transferring an *image model* from 2D space to 3D space is still very much an unsolved problem. In this chapter, we present a possible solution to this problem. We model the 2.5D pseudo-solid texture models trying to satisfy some deterministic 3D spatial features, where key features exist on surfaces of volume objects. We will demonstrate in later chapters of this thesis that our projective texture model produces superior results in these areas by providing a flexible rendering framework.

The additional contribution of our generic volume object model is that it provides a universal model for integrating 2D images and 3D volume datasets, giving a universal solution to many DSOR based applications. It challenges many other visualisation and illustration algorithms and generates more realistic results. Moreover, it provides the flexibility and possibility of integrating various computer graphics and volume graphics applications. The high quality visual effects and realistic appearance prove its efficacy.

Chapter 4

Multi-Dimensional-Scaling Models (MDS)

Point sampling and point-based rendering techniques are becoming basic research in volume graphics, for instance, moving from grid-based volume graphics to point-based volume graphics [8], and rendering multi-resolution point-based surfaces [7]. We believe that manipulating point clouds for texturing volume datasets can become a new bridge connecting volume graphics and image-based texture models, as we will demonstrate in this chapter.

In order to improve the shear effect of the projective texture model, we here present a graph model which preserves the neighbourhood relationships of the point clouds within volume objects. In order to further smooth the point clouds, the classic multi-dimensional scaling (MDS) using novel shortest-path proximities that is based on the graph model is also introduced.

By mapping 3D points into a 2D flattened (Euclidean) domain the methods presented can both flatten the projected 3D surfaces of volume objects (onto intermediate templates) and preserve local geometrical features on the surface (level sets) of volume objects.

We will also discuss point clouds tangling and flipping in MDS methods. In this chapter we will introduce:

- A novel method to construct point clouds of volume objects by sampling the positions of level sets using projective spherical models.
- A point clouds smoothing method using the MDS method, which allows us to

flatten the intermediate template and thus match textures to the shape of the object.

This is done by constructing a novel projected (plenoptic) graph model that is composed of these probed 3D data points (node) and the Euclidean-distances between each neighbouring 3D points (edges), the algorithm we call the *plenoptic shortest-path MDS*.

- A novel method to improve shear effects, by overlaying texture images onto the flattened intermediate templates.

  Our method of flattening 3D volume data sets for the applications of texture mapping and annotation focuses on linear dimensional reduction, graph layout, preserving neighbouring relationships among 3D point clouds, with no necessity to adjust any parameters.

4.1 Introduction: Shear Effects

Motivated by the observations of the importance of realistic visual appearances of volume objects, we presented an imaged based approach to texture mapping volume datasets [39, 40] in Chapter 3. The method is based on a projective pseudo-solid texture model and semantic constraints. A rendered intermediate template for texture warping is needed.

The rendered intermediate template is based on projective mapping. Therefore, texels will be smeared over a relatively large area if they are projected onto a surface which is almost parallel to the direction of projection, the phenomenon that is referred to "shear effect" in this thesis.

In this chapter, we are trying to introduce a mesh-less model for texturing volume objects and reduce shear effects, that is, directly manipulating a point cloud rather than constructing a mesh model as an intermediate step for texture mapping and annotation.

4.2 Related Work

Many existing texture models were designed to support mesh-based CG characters. Here we draw our inspiration mainly from texturing or annotating volume objects

without mesh models. Therefore, the techniques which are directly relevant to manipulating points cloud for texturing volume objects are particularly interesting to us.

In this chapter, we focus on mesh based surface flattening methods [38], which provide positioning control for texture mapping. Here, we discuss the tangling and flipping of mesh models.

Tangling and flipping

Preserving the quality of moving mesh grids and improving the efficiency of mesh smoothing algorithms are common problems in many practical applications. Unfortunately, we can often see a few lines of mesh models cross the other mesh lines after mesh deformations. Then, this is just that we called "mesh tangling and flipping" in this chapter.

To improve the accuracy and efficiency of mesh deformation, the mesh grid can be regenerated according to the salient features of the local regions. However, since these local regions can change with time, the local regeneration or refinement of the mesh grid can become extremely computationally demanding. Under such circumstances, Bochev et al. developed and analyzed the method that both uses fixed mesh grid structures and distributes grid nodes according to a given analytic weight function of the spatial variables [119]. By defining the appropriate weight functions, they demonstrated that their methods can accurately reposition the nodes and do not tangle the mesh.

The performances of mesh moving algorithms using fixed grid structures are of particular interests to us: as Bochev et al. pointed in [119], such methods do not require complicated data structures, since the reference domain is not changed; cost of moving the grid is limited to computation of new physical coordinates of the grid points. Therefore overall efficiency of moving grids can be significantly improved.

Bechev et al.'s experimental results indicate that their methods do not tangle the mesh after moving operations. However, it is worth pointing out that we are currently dealing with a different problem. Here we are trying to flatten a point cloud from 3D space into a 2D plane. Therefore there is no explicit mesh grid that can be used.

Mesh quality for moving meshes in 2D and 3D unstructured mesh models were also discussed by Berzins et al. [120]. The solutions to the problems of maintaining mesh quality of unstructured triangular and tetrahedral meshes are considered. They investigate the anisotropic properties of the atmospheric dispersion mesh model. Tangling was discussed for moving the unstructured mesh models in 2D or 3D space indepen-

dently. However, this is also differences with our target which focuses on preventing tangling while we flatten mesh models from 3D space into 2D plane.

4.3 Principle of the Algorithm: MDS Models

4.3.1 Background: classic multi-dimensional-scaling (MDS)

Given a discretely sampled object representation (level sets), we would like to construct a spatial relationship representing the 3D geometrical configurations of these sampled points. The abstracted spatial relationship should map the DSOR representations (point clouds, level sets) into an analytic and function domain.

Starting from the discretely sampled object representations (DSORs), which lack functional information, we need to use *proximities* among DSOR level sets and then output the spatial map using the proximity measurements. In fact, *multidimensional scaling, MDS*, is the solution to this problem [121].

- Proximities:

  A proximity is a measure that indicates how similar or how different the two objects or two elements of an object to be. In fact, any kind of information such as geometrical, topological or semantic, can be regarded as a measure of proximity among different individuals.

- Multi-dimensional scaling (MDS):

  Multi-dimensional Scaling (MDS) refers to a class of techniques. The input of these techniques are proximities amongst any kind of objects (or, elements of an object). The output is the spatial relationship reflecting the hidden structures among the data set. Such a spatial relationship is often represented as a map, which demonstrates the geometric configuration of points.

In this thesis, the output of MDS is a 2D Euclidean plane representing the given proximity measures. The input is the proximity measure matrix whose element is the distance (proximity measure) between each 3D point in the level sets. Given a proximity measure matrix, M, then the 2D flattened configuration can be calculated via *Classical Scaling*, a direct and metric MDS method. In metric MDS, the original

distance (or proximity) matrix is approximated. Non-metric MDS deals with data in which the order of the distances must be preserved [122].

The classical MDS method can be defined as follows [38]:

Given a set of 3D points $\{\bar{p}_k, k = 1, 2, ..., n\}$, and proximities $\delta(k, l)$ between points $\bar{p}_k$ and $\bar{p}_l$, we need to find a set of 2D data vectors $\hat{\bar{p}}_k$, $\hat{\bar{p}}_k \in \Re^2$, that have Euclidean distances $\{\hat{d}(k, l)\}$ which approximate the proximities $\delta(k, l)$ well. Here, the 3D points within volume objects are the 3D positions of point clouds sampled by firing tracing rays using the spherical model, $\bar{p}_k = (X_k, Y_k, Z_k)$.

Define the matrix $P_{n,3}$ to represent the (X, Y, Z) positions of n points in 3D Euclidean space. The square proximity distance between point $\bar{p}_i \in P_{n,3}$ and $\bar{p}_j \in P_{n,3}$ is defined as:

$$d_{ij}^2 = (proximity(\bar{p}_i, \bar{p}_j))^2 \tag{4.1}$$

Let M be a matrix whose entries are defined by $M_{ij} = d_{ij}^2$, $i, j = 1, 2, ..., n$, then the classic scaling MDS method can be calculated using eigenvector decomposition:

$$Double\ centring\ and\ normalisation : B = -0.5 * J * M * J \tag{4.2}$$

$$Eigenvector\ decomposition : [Q, L] = eigs(B, 2, 'LM') \tag{4.3}$$

$$newx = sqrt(L(1, 1)). * Q(:, 1); \tag{4.4}$$

$$newy = sqrt(L(2, 2)). * Q(:, 2); \tag{4.5}$$

$$J = I - \frac{1}{N} L \cdot L^T; \tag{4.6}$$

where J is the centring matrix which moves the origin of matrix M onto the centre of its mass. Matrix I is an identity matrix and vector L is the vector of ones (1's). In Equation (4.3), the centred and normalised proximity matrix B is approximated by the matrix L whose rank is 2. The equation is solved in the least square sense. In Equations (4.4) and (4.5), the flattened 2D coordinates of the i-th 3D point are $(newx_i, newy_i)$, where $i = 1, 2, ..., n$. Note that the flattened 2D coordinates are obtained by multiplying the two largest positive eigenvalues with their associated

eigenvectors.

4.3.2 Geodesic-based MDS models

Grossmann et al. [122] used minimal geodesic distances between points on the surface to construct the proximity matrix M. By constructing the geodesic distance on a 3D surface, their algorithm can be described as follows:

(1) Create proximity matrix (squared geodesic distance matrix):

$$D = \{d_{ij}^2 = (geodesic\_distance(x_i, x_j))^2, i, j = 1, 2, ..., n\} \tag{4.7}$$

(2) Calculate the estimated scalar product matrix:

$$\hat{B} = -0.5JDJ \tag{4.8}$$

(3) Calculate the eigen-decomposition of $\hat{B}$ up to rank 2:

$$\hat{B} = Q_2 \Lambda_2 Q_2^T \tag{4.9}$$

where Λ_2 and Q_2 are 2×2 and $n\times2$ submatrices.

(4) The flattened 2D coordinates matrix is:

$$\hat{X} = Q_2 \Lambda_2^{\frac{1}{2}} \tag{4.10}$$

The above classical scaling method was employed for flattening 3D surface points. For our interests, the unique features of the method are:

- direct operations on voxels without the necessity of constructing a mesh model as an intermediate step;
- optimal estimations of Euclidean distances of edge-length e between vertexes v on volume data;
- global preservation of minimal geodesic distances of paths from the source vertex to the destination vertex on the graph model $G(v, e)$;
- computational efficiency.

Floating and isolated voxels:

In Equation (4.7), there are two crucial steps for estimating the minimal geodesic distances between 3D points on voxel based surfaces:

(1) Representing the surface as a weighted graph $G(v, e)$:

$$voxel \Longleftrightarrow v : vertex, \quad link \Longleftrightarrow e : edge \tag{4.11}$$

(2) The weight of an edge depends on the three link types: *direct*, *minor diagonal*, and *major diagonal*. The unbiased, minimum Mean-Square-Error length estimation for a 3D curve is:

$$\begin{aligned}\hat{L} =& 0.9016 * Number(direct\ links) + 1.289 * Number(minor\ diagonals) + \\ & 1.615 * Number(major\ diagonals)\end{aligned} \tag{4.12}$$

where function $Number()$ calculates the numbers of the different link types (N_1, N_2, N_3, as shown in Figure 4-1) between neighbouring voxels on the path.

Figure 4-1 shows the three link types in a 26-directional 3D chain code [122].

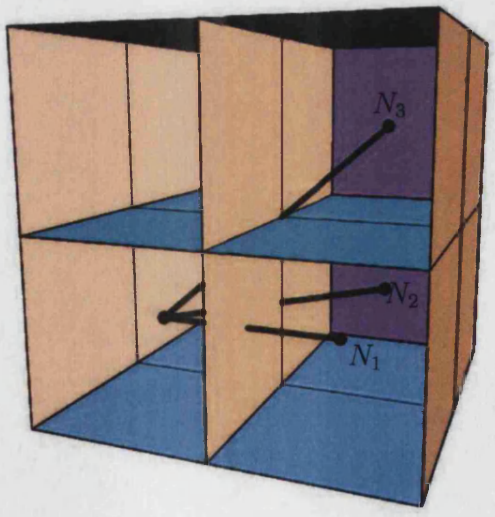


Figure 4-1: Link types in a 26-directional 3D chain code [122]: N_1: direct link (parallel to one of the main axes), N_2 : a minor diagonal link, and N_3: a major diagonal link.

The design of 3D length estimators must be based on theoretical analysis of chain code probabilities in three dimensional chain encoded lines [123]. The contributions of the above equation, Eq.(4.12), are not only the minimum RMS estimators but also the predictions of the number of direct, minor diagonal and major diagonal links in the chain code of a 3D line.

Given the probabilities of a 3D chain code of a 3D line, the crucial problem of calculating the minimal geodesic distance between source position and destination position on a 3D voxel based surface is how to let the geodesic trajectory pass through the "bridging" voxels on the path, as demonstrated by Grossmann et al. [122] and Kiryati and Székely [124].

Unfortunately, since DSORs lack geometrical, topological and semantic information, we cannot guarantee such "bridging" always exists. There is a high possibility that the 3D point clouds are isolated level sets, rather then just the discretely sample positions on a *connected* surface. So, we cannot be sure that there is at least one path between any pair of two voxels on the surface. In this chapter, we deal with the following special cases:

- We do not require the existence of a path from source voxel to target voxel for estimating minimal geodesic distance.
- We start from the primary DSOR sampled positions, which lack topological, geometrical information. These 3D positions are theoretically isolated points, without any explicit connecting constraints.
- Grossmann et al.'s work started from an explicitly defined connected surface, represented by either 3D point clouds or voxels. However, in a reverse sense, our work in this chapter resides on isolated level sets to represent DSOR-based surfaces.
- In addition, we use plenoptic models to fire tracing rays to construct 3D point clouds within volume objects. The plenoptic models are also used to construct the graph model to calculate shortest paths for MDS.

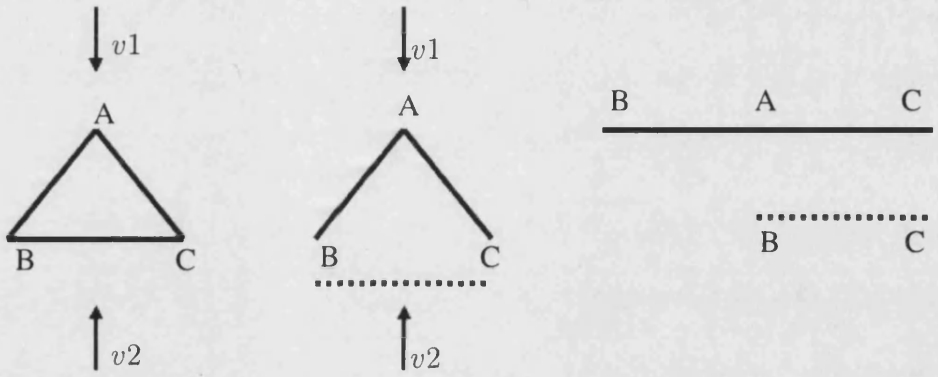
4.3.3 Geodesic distance ambiguity

As discussed in Chapter 3, we use projective texture models to generate intermediate templates. The texture projections include planar projection, spherical projection, cu-

bic projection and cylindrical projection. The volume object can thus be viewed from any position in 3D space.

Given two voxels on a volumetric level set (without an explicitly constructed mesh model), if we view the object from one viewpoint, then the geodesic path between these two voxels might be different to one viewed from the opposite viewpoint. The two shortest paths run through different terrain on the different surfaces (level sets) facing the two opposite viewpoints.

If we try to flatten the enclosed surface using these two different shortest geodesic distances, then the two flattened surfaces will be different to each other. We here refer to this phenomenon as the ambiguity of shortest paths of point clouds. An example is given in Figure 4-2.



(a) (b) (c)

Figure 4-2: Geodesic distance ambiguity between opposite viewpoints: (a) The triangle ΔABC is observed from two different viewpoints, $v1$ and $v2$. (b) The geodesic ambiguity exists in a closed structure. If viewing from $v1$, the geodesic distance (geodesic-path) is $\overline{BAC}$. If viewing from $v2$, the geodesic-path is $\overline{BC}$. (c) Different distances of flattened vertexes of the triangle: $\overline{BAC}$ is the flattened geodesic-distance viewed from the viewpoint $v1$. $\overline{BC}$ is the flattened geodesic-distance viewed from the viewpoint $v2$.

Given the triangle ΔABC, the minimal distances (edge lengths) between the three vertexes are: $\overline{AB}$, $\overline{BC}$, $\overline{AC}$. The edges are composed of neighbouring voxels. If we observe the triangle from the viewpoint $v1$ and would like to unfold the vertexes B, C

and A, the estimated minimal geodesic distance should be $\overline{BAC}$, which approximates the flattened Euclidean distance from vertex B to C to A. Unfortunately, the shortest geodesic distance might also be $\overline{BC}$, represented using the dotted line.

In order to flatten the points A, B, and C, we need to cut the connection between B and C, either by physically deleting elements on the line BC, or by logically deleting the linking elements in the graph model $G(v, e)$. We refer to the multiple possibilities of minimal geodesic distance between the same pair of vertexes as the *geodesic distance ambiguity.*

This problem becomes worse in the circumstance of flattening a DSOR data set. First, cutting a connection (voxels) needs at least topological and geometrical information; however, we do not have such information at hand. Second, we cannot keep a record with both observation configurations and associated geodesic paths, since to do so needs a full understanding and exploration of the volume data.

These two weak points become the main obstacles for constructing a spatial relationship reflecting the hidden structures of 3D point clouds. Even worse, as we will explain later, tangling and flipping are inevitable due to such geodesic-distance ambiguity.

4.3.4 Euclidean-distance based classic MDS models

Euclidean-distance based proximity matrix

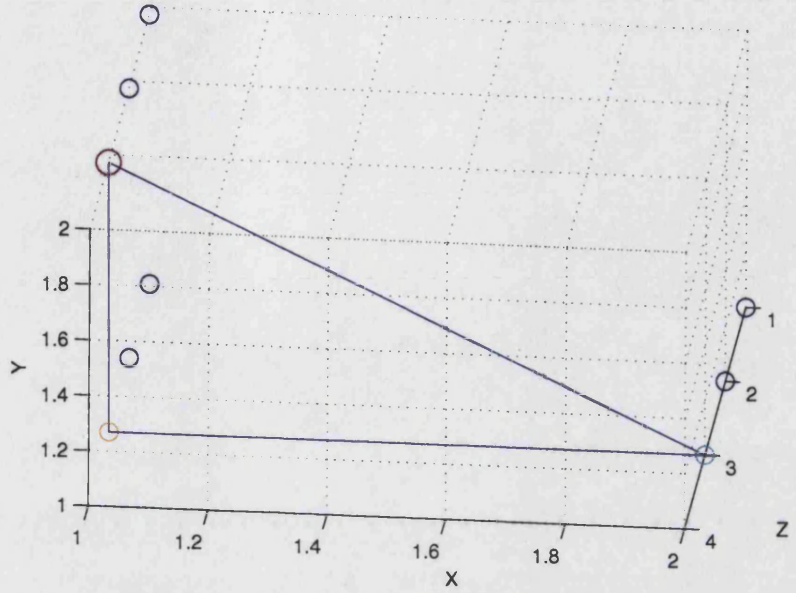
Given a matrix $P_{n,3}$ to represent the (x, y, z) positions of n points in 3D Euclidean space; the square Euclidean distance between point $\bar{p}_{i,a}$ and $\bar{p}_{j,a}$ is defined as:

$$d_{ij}^2 = \sum_{a=1}^{3} (\bar{p}_{i,a} - \bar{p}_{j,a})^2 \tag{4.13}$$

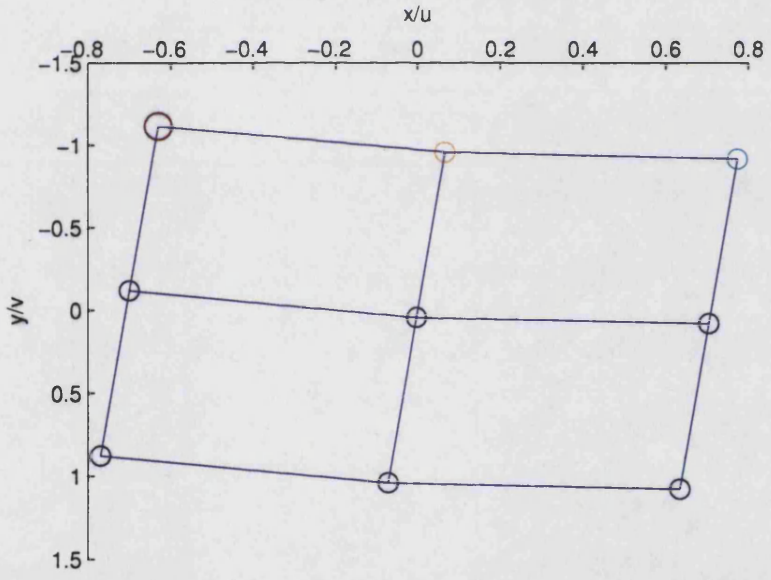
The proximity matrix can be constructed by using such square Euclidean-distance proximities, $M_{ij} = d_{ij}^2$. After constructing proximity matrix M, we can use the classic scaling MDS (Equations (4.2) to (4.5)) to flatten the 3D point clouds.

In Figure 4-3, figure (a) shows a synthesised 3D 3x3 point data set and figure (b) shows the 2D positions of the flattened data set.

Here we use the Euclidean-distance based classic MDS method. In this example, we can see that the 3D points can be flattened without any flipping and tangling of the 2D positions.

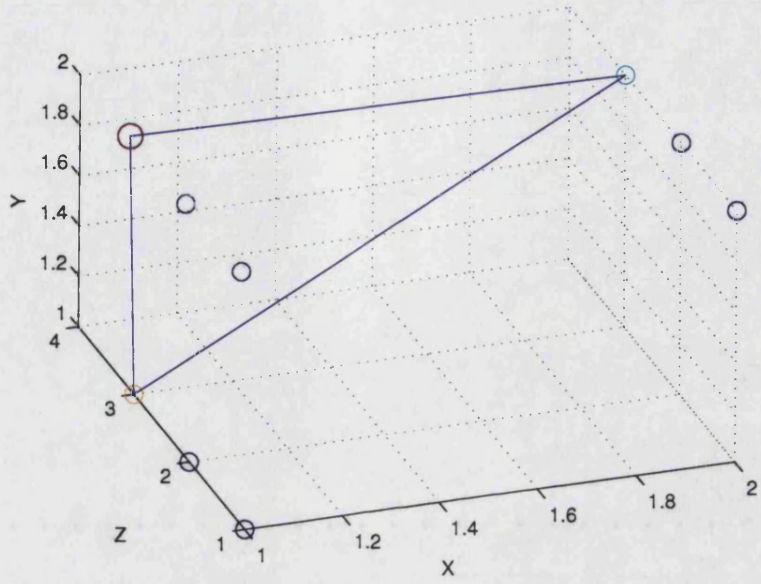


(a) A synthetic 3D data set.

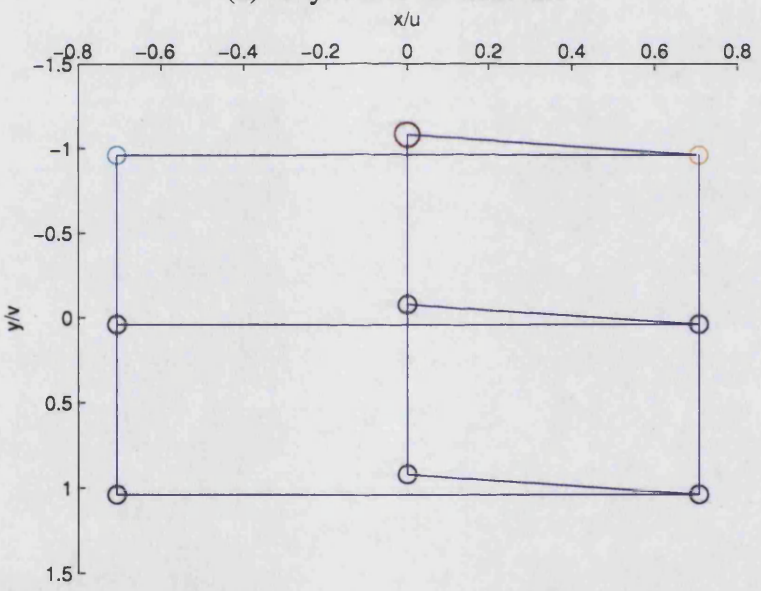


(b) Flattened 3D point set using MDS

Figure 4-3: Flattening 3D point sets without flipping and tangling. The flattened 2D positions (red, yellow and blue) preserve the neighbourhood relationships.



(a) A synthetic 3D data set.



(b) Flattened 3D point set using MDS

Figure 4-4: Flipped and tangled 3D point sets flattened using MDS. In figure (b), the lines cross the others in the flattened configuration.

The 3D points in red, yellow and blue, are the first, the second, and the third point within the 3D point set. The first (red) 3D point is neighbouring the second (yellow)

3D point, and the second (yellow) is neighbouring the third (blue) 3D point.

The flattened 2D positions preserve such neighbourhood relationships. In addition, an example of the tangled 2D positions (lines cross in the flattened plane) is given in Figure 4-4(b).

Interactive data manipulation

Interactively adjusting MDS configurations and manipulating the proximity matrix using weighting functions are common techniques in multidimensional scaling analysis [125, 126]. Common techniques include, for instance, power transformations for metric MDS, distance transformations, group-dependent MDS, etc. These techniques benefit the data visualisation in MDS. However, they also tell us that a single MDS techniques cannot cover the variety of data visualisation problems. In particular, as we will now explain, the existence of flipping and tangling of 2D positions, is a major obstacle for annotating volume objects using 2D texture images.

Flipped and tangled point clouds

In Figure 4-4, if we try to flatten the 3D data set shown in figure (a), the flattened 2D positions become flipped and tangled, i.e, the right side of the quadrilateral model is flipped onto the left side of the quadrilateral model. In other words, as shown in figure (b), the neighbourhood status, i.e, from red to yellow to blue, becomes the relationship starting from blue to red to yellow. So, we refer to the flipping and tangling shown in figure (b) as the phenomenon that lines cross the other lines in the flattened configuration.

As shown in Figure 4-3(a), the Euclidean-distance between the first (red) and the third (blue) 3D points is greater than the distance between the first (red) and the second (yellow) points, and it is also greater than the distance between the second (yellow) and the third (blue) points. In Figure 4-4(a), the Euclidean-distance between the first (red) and the third (blue) 3D point is less than the distance between the second (yellow) and the third (blue) 3D points.

From Figure 4-3 and Figure 4-4, we can see that estimating an appropriate proximity matrix for MDS methods plays a critical role in analysing hidden structures of high-dimensional data sets. We noticed that directly calculating the minimal geodesic distances running along the 3D voxels on the surface of 3D volume objects is not ap-

propriate to us, since there is no guarantee of the existence of the minimal geodesic distance between each pair of 3D points. Instead, we can calculate the shortest-path length to approximate the geodesic distance between each pair of probed 3D points.

4.3.5 Plenoptic graph model and shortest-path MDS

Neighbouring relationships of 3D point clouds

As discussed in the previous subsection, in order to estimate an appropriate proximity matrix for MDS methods, we can calculate the shortest-path length to approximate the geodesic distance between each pair of probed 3D points.

This can be done by constructing a novel projected (plenoptic) graph model which is composed of these probed 3D data points (nodes) and the Euclidean-distances between each pair of neighbouring 3D points (edges), the algorithm we call the *plenoptic shortest-path MDS*. Given a pair of 3D points probed by two tracing rays, if the origins of these two tracing rays are neighbouring positions on plenoptic surfaces, then these two 3D points are defined as neighbouring 3D points. So, we refer to the neighbouring relationships of 3D points as the neighbourhood status of the origin positions of the tracing rays used to probe the 3D positions of these 3D points.

To our knowledge, texture tangling have not been discussed in the MDS applications. The conventional MDS implementation is prone to uncertainties, i.e, "*artefacts in point configurations*" and "*local inadequacy of the point configurations*" [125]. Traditional solutions to these problems are: first, "*diagnostics for pinning down artefactual point configurations*", and, second, "*restricting MDS to subsets of objects and subsets of pairs of objects*".

In this chapter, tangling (lines crossing in flattened configurations) originates from the differences between the neighbouring status of sampling positions (used as the origin positions to fire tracing rays) on plenoptic surfaces and the neighbouring status of the 3D points actually probed. Obviously, there is no guarantee of the coincidence between these two.

A Euclidean-distance based proximity matrix just constrains the spatial distance between each pair of 3D point clouds. There is no neighbouring relationship among these 3D points. In contrast, a shortest-path proximity matrix introduces neighbouring constraints by constructing the graph model. The graph model not only constrains the direct neighbouring status using shortest-path, but also constrains the neigh-

bouring status of the 3D points on the shortest-path. In other words, if the path $(v_0, ..., v_i, ..., v_j, ..., v_n)$ is a shortest path from v_0 to v_n, then the shortest path from v_i to v_j must be a fragment on this path.

Graph layout and local information

As previously discussed, graph layout and local information about the original data set were employed in MDS applications to improve the visibility and the accuracy of interpretation of MDS configurations. In particular, using local neighbourhood information to construct a global low-dimensional configuration of manifold data is a basic research topic [127]. Employing ideas of graph layout techniques such as parameterised based energy functions, and the novel meta-criterion, Chen and Buja's local MDS (LMDS) could both create faithful embeddings and provide a measurement of the local adequacy of embeddings [127].

LMDS ties together three areas: nonlinear dimension reduction, graph layout, and proximity analysis. However, it still has tuning parameters to generate a robust embedding configuration for noise data set. Therefore, our consideration of flattening 3D volume data sets for the applications of texture mapping and annotation focuses on linear dimensional reduction, graph layout, preserving neighbouring relationships among 3D point clouds, and no necessity to adjust any parameters.

In particular, while LMDS focuses on providing local continuity based meta-criterion to proximity estimation, our algorithms focus on local continuity (neighbourhood) based anisotropic parameters. However, as we will explain in the next chapter, these parameters will be further used in linear weighted Laplacian smoothing.

4.3.6 Proximity matrix: shortest paths

In this subsection, we use a shortest-path based proximity matrix to flatten the 3D point cloud.

Given an $n \times n$ probed 3D point cloud, we use the graph $G(V, E)$ to model the point cloud. $V = \{1, 2, ..., n \times n\}$ is the node set and $E = \{e_{i,j}, i \; is \; neighbouring \; j\}$. Then the shortest-path proximity matrix M can be described as follows:

$$M = \{m_{i,j} = shortest\_path(i, j), i \in V, j \in V\} \tag{4.14}$$

The implementation of the shortest-path function is the famous Dijkstra algorithm [128], described as follows:

We refer to the weight of a path as its length, the minimum weight of a (u, v)-path will be distance from vertex u to vertex v, denoted by $d(u, v)$.

Given each vertex v, its label $l(v)$ is an upper bound on the distance $d(u_0, v)$ between the starting vertex u_0 and an intermediate vertex v. Initially $l(u_0) = 0$ and $l(v) = \infty$ for $u_0 \neq v$.

First, insert all intermediate vertices v onto set $\bar{S}_0$ with their labels $l(v) = p(u_0, v)$. Delete the vertex with minimum label in set $\bar{S}_0$ and add it to the solution set S_0. Using an iterative process, the label $l(u_i)$ of the newly inserted vertex u_i in set S_i and the lables $l(v)$ in set $\bar{S}_j$ can be updated using the following equation:

$$l(u) = d(u_0, u_i) \tag{4.15}$$

$$l(v) = MIN_{u_{i-1} \in S_{i-1}}\{d(u_0, u_{i-1}) + w(u_{i-1}, v)\} \; for \; v \in \bar{S}_i \tag{4.16}$$

where $w(u_{i-1}, v)$ is the sum weight from u_{i-1} to v.

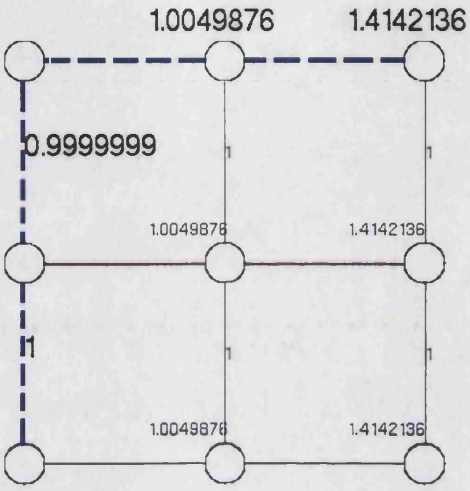
Repeat the above iterative process. If u_i equals v_0, then stop. The shortest path between vertex u_0 and vertex v_0 is saved in label $l(v)$.

As shown in Figure 4-5, we use the shortest-path based MDS to flatten the same 3D point clouds shown in Figure 4-4. Figure 4-5(a) is the graph model of the data set shown in Figure 4-4(a). The shortest-path from bottom-left corner to top-right corner is marked in blue (dashed lines). The weights between each pair nodes are also given. The numbers are the weights of the edges. Figure 4-5(b) shows the flattened 2D positions of the 3D data set.

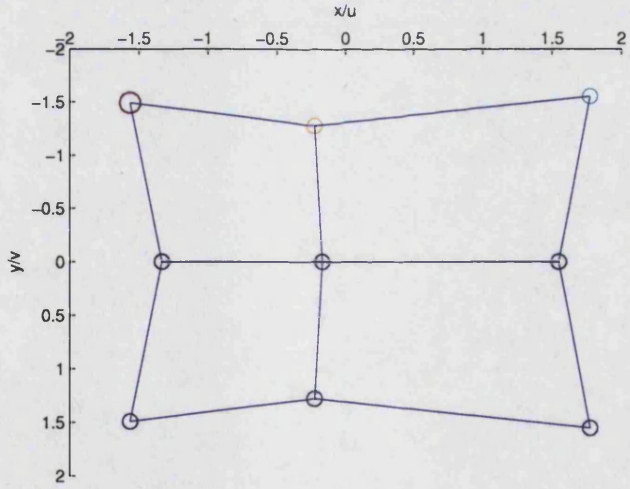
Clearly, by constructing the spherical graph model which preserves the neighbourhood relationships between each pair of nodes, and employing the shortest-path proximity matrix, the flipping and tangling are eliminated. Note that, first, by using the graph model of the data set shown in Figure 4-5(a), we can actually eliminate the ambiguity of calculating geodesic distances; second, the neighbourhood relationships (connectivity) between each pairs of points are preserved. Therefore, the flattened configuration does reflect the preserved neighbourhood relationships.

Note that eliminating the ambiguity of geodesic distances and preserving the neighbourhood relationships are not sufficient conditions yet to flatten point clouds without tangling and flipping. The lengths of shortest path between boundary points also play

critical roles. In general, in order to embed all the flattened internal nodes within the boundary that is composed of flattened boundary nodes, the length of the shortest path of any pair of internal nodes must be a fragment on a shortest path of the pair of boundary nodes.



(a) The shortest path from the bottom-left node to the top-right node.



(b) The flattened 3D point set using shortest-path MDS

Figure 4-5: Flattened 3D point sets using shortest-path MDS, which is based on the shortest-path based proximity matrix M.

By constructing the novel *plenoptic shortest-path MDS*, we present the solution to

the problem of the ambiguity of calculating geodesic distances. We also preserve the neighbourhood relationships of point clouds using the spherical graph model. More examples of the novel model which can be used to improve the quality of flattened configurations will be further given in the following sections.

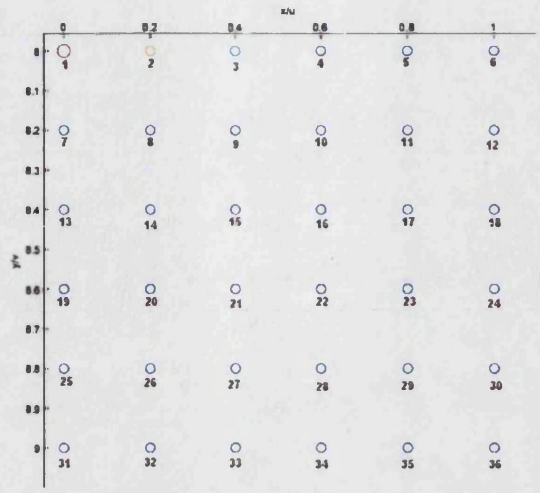
4.4 Global and Local Properties of Plenoptic MDS

Intermediate templates provide the texel positioning control for texturing volume objects. It is a flattened plenoptic surface whose pixels are rendered using volume rendering algorithms. Ideally, the intermediate template itself should be a flattened surface of volume objects, so that a texture image can be overlaid onto it directly.

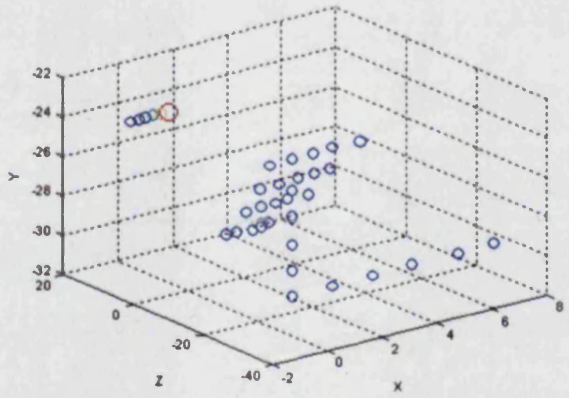
In previous methods the intermediate template is rendered using sampling positions on plenoptic surfaces. For each of the sampling positions, tracing rays probe 3D positions within the volume dataset (iso-values, intensities, distance fields, etc.) So we know the sampling points both in the 3D space and in the 2D grid coordinates. As shown in Figure 4-6(a), the 6x6 positions are the sampling positions on the plenoptic surface, which are used as the origin positions to fire tracing rays. The probed 6x6 3D points cloud is shown in Figure 4-6(b). Then we flatten the 3D point cloud using the Euclidean distance based matrix, M, (a classic MDS method in practice). The flattened points are shown in Figure 4-6(c). The neighbouring points are connected using the connection relationship of adjacent sampling positions on the plenoptic surface, shown in Figure 4-6(a). Unfortunately, the flattened points becomes twisted and tangled.

The 3D points shown in Figure 4-3(a) are from a synthetic data set. Now the 3D points shown in Figure 4-6(b) are probed using tracing rays fired at the sampling positions on a plenoptic surface. Sampling positions shown in Figure 4-6(a) are the positions on the plenoptic surface (u, v) and the positions on the intermediate template (x, y).

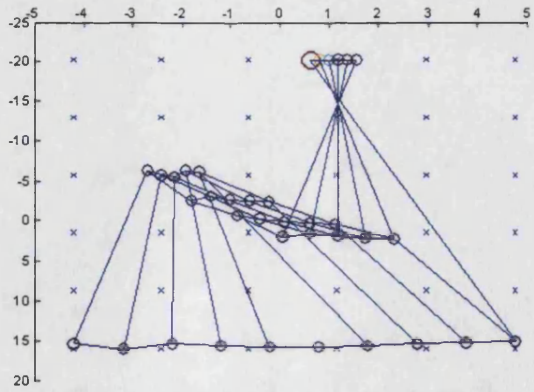
Note that, first, we do not construct mesh models for volume datsets; second, we do not use the geodesic distance to construct the matrix M in MDS. Different paths' geodesic distances between two vertexes on a mesh model will be equal to each other on the flattened plane [38]; however, such an assumption is not further valid to a points cloud. In fact, geodesic distances will be different to each other when they run different paths in 3D space.



(a) 6x6 spaced sampling positions on a plenoptic surface ((u, v)) and on an intermediate template ((x, y)).



(b) 6x6 3D cloud points probed by fired tracing rays.



(c) Tangled flattened surface using MDS.

Figure 4-6: Euclidean based MDS: (a) 6x6 sampling positions on a plenoptic surface and on an intermediate template. (b) Sampled 3D points. (c) Tangled and flipped flattened point cloud.

As previously discussed, Euclidean-distance based proximity measurements only guarantee the spatial relationship between pairs of 3D point; no neighbour constraints exist. Therefore, for both geodesic-distance based classic MDS and Euclidean-distance based classic MDS, these is no guarantee of preventing such flipping and tangling behaviour. So, the novel neighbourhood relationships on plenoptic surfaces are introduced into MDS by constructing the graph model of sampling positions on plenoptic surfaces.

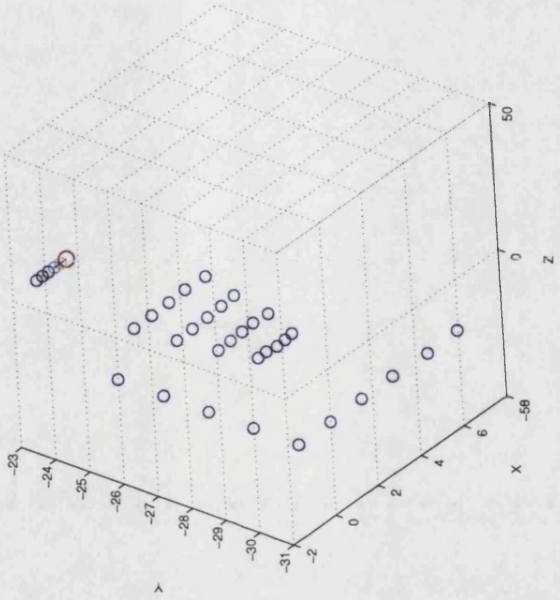
As shown in Figure 4-7, we use the shortest-path based MDS method to flatten the 3D point cloud. Figure (a) shows the same 3D point cloud viewed from a different view point. Figure (b) shows the flattened point cloud. The close up of the area within the red box is shown in Figure 4-8(a). We can see that the point 5 and the point 6 are smoothed without flipping. The close up of the area within the green box is shown in Figure 4-8(b). We can see points 26, 27, 28, 29, 30 are smoothed with flipping and tangling.

- Global and Local Properties of MDS:

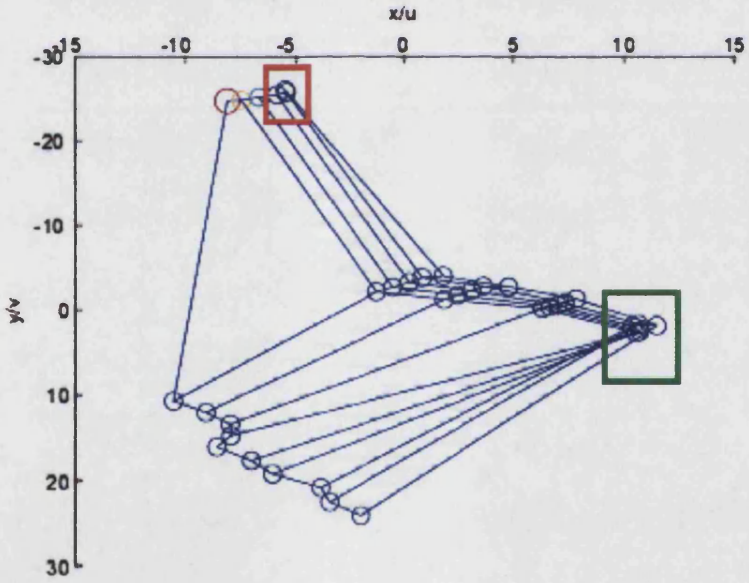
It is well known that [125] "*The global shape of MDS configurations is determined by the large dissimilarities; consequently, small distances should be interpreted with caution: they may not reflect small dissimilarities.*"

Points 26, 27, 28, and 29, located within a small 3D space, construct the so called *minimal local structures*. Since classic MDS is a principal component (larger proximities) dominated minimisation process, the less important component (small proximities) cannot really do their contribution to the global based configuration.

Note that: first, a minimisation process without large proximities often does not generate meaningful global configuration in MDS; second, attempts at integrating local structure to model global structure are often not successful. In practice, truncation and weighting functions are interactively used to explore local data structures using MDS methods.

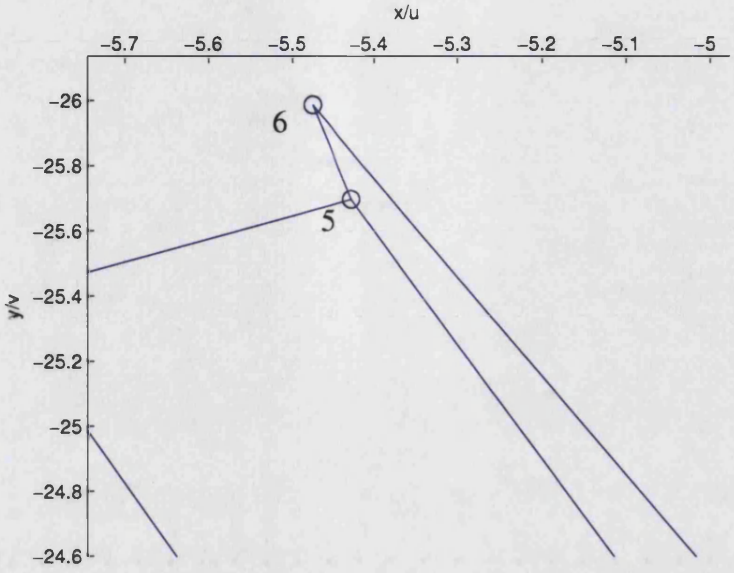


(a) The same 3D point cloud shown in Figure 4-6(b)

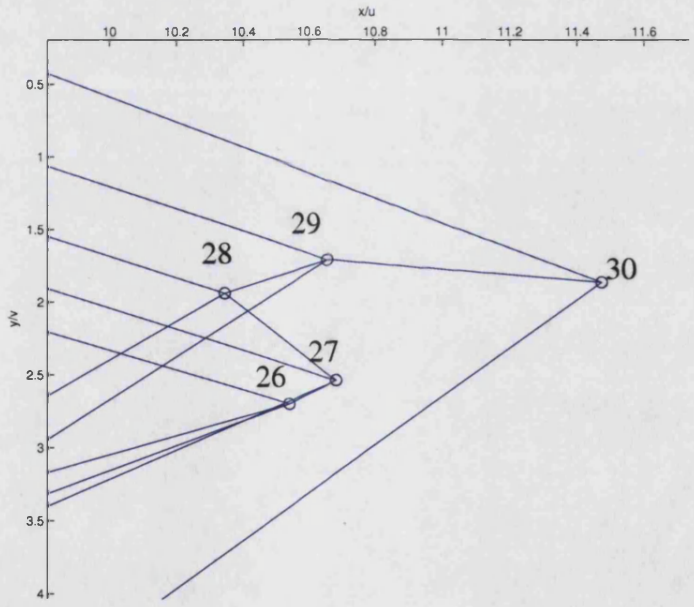


(b) Flattened point cloud using shortest-path based MDS.

Figure 4-7: Shortest-path based MDS: Flattened 3D point cloud using shortest-path proximities. (a) The same 3D points cloud shown in Figure 4-6(b), viewed from a different viewpoint. (b) The flattened configurations using shortest-path proximity matrix and classic MDS. The close-ups of the areas within red rectangles are given in Figure 4-8.



(a) Close-up of the flattened 5th point and 6th point: no tangling



(b) Close-up of the flattened points, tangling of small point cloud clusters.

Figure 4-8: Shortest-path based MDS: Tangled flattened point cloud in small areas enclosed in the red and the green rectangles shown in Figure 4-7(b). In (b), the flattened points, 26th, 27th, 28th, 29th, are tangled, within the small area.

4.5 Experimental Assessment

We use CTHead to test our algorithms. As shown in Figure 4-9, the point cloud consists of 21x21 3D positions probed using a spherical projective texture model. The 3D coordinates of the positions are the terminating positions of the tracing rays, which are based on direct volume rendering (DVR).

CTHead: discretely sampled point clouds

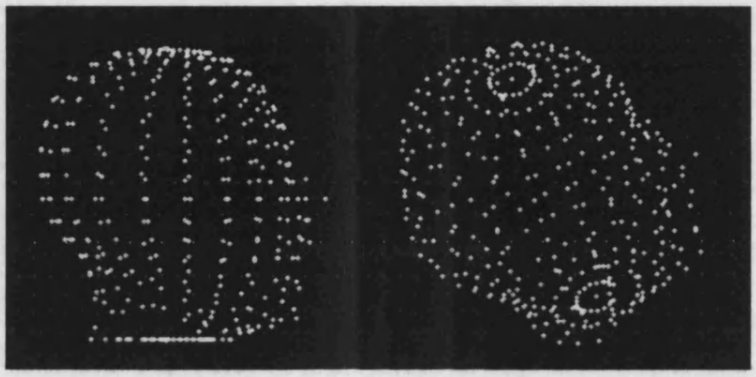


Figure 4-9: Point cloud of CTHead: front view and top view

Given the probed 3D positions, we first flatten these 3D points using the classic MDS method which is based on a Euclidean-distance based proximity matrix. The configuration is shown in Figure 4-10. We see that the shape of the flattened 2D configuration does not actually match to a flattened shape of the enclosed surface of the volume CTHead data.

The Shepard diagram (scatterplot) is shown in Figure 4-11. The vertical axis, $Euclidean_y$, is the Euclidean distance between each pair of flattened 2D positions. The horizontal axis, $Euclidean_x$, is the Euclidean distance between each pair of 3D points. As shown in Figure 4-11, the Euclidean distances between each pairs on the flattened planar configuration are less than the Euclidean distances between the pairs of the original 3D points. We refer to this observation as the "squash" version of the 2D configuration.

Euclidean-distance MDS flattening

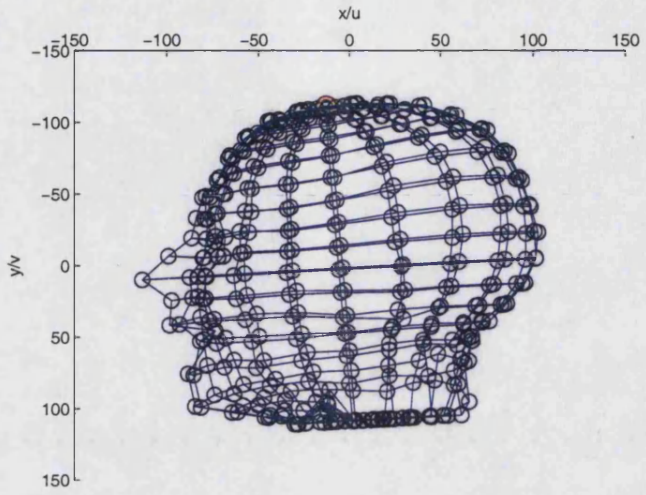


Figure 4-10: MDS-Flattened point cloud of the CTHead using a Euclidean-distance proximity matrix.

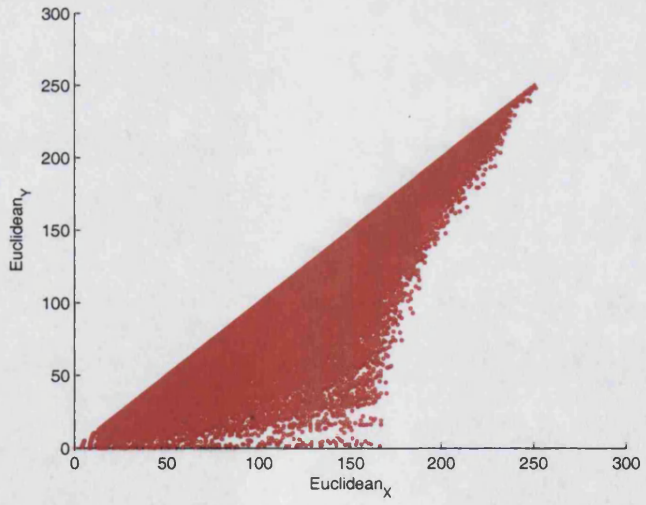


Figure 4-11: The Euclidean distance on the 3Dsurface, $Euclidean_y$, versus the Euclidean distance on the flattened planar configuration, $Euclidean_x$. The data corresponds to the CTHead point cloud shown in Figure 4-10.

Shortest path based MDS flattening

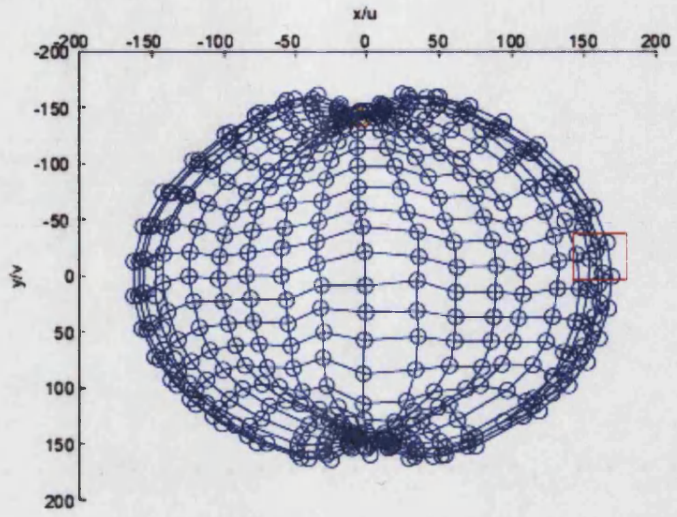


Figure 4-12: Flattened point cloud of the CTHead using a shortest-path proximity matrix. The close-up of the area within the red box is given in Figure 4-14.

Given the probed 3D positions, we now flatten these 3D points using the classic MDS method which is based on a shortest-path proximity matrix. The configuration is shown in Figure 4-12. We can see that points are mostly flattened in the 2D configuration. The 3D points are flattened according to the 3D shape of the surface (level sets) of the CTHead data.

The Shepard diagram is shown in Figure 4-13. The horizontal axis, $Euclidean_x$, is the Euclidean distance between each pair of flattened 2D positions in the configuration. The vertical axis, $Euclidean_y$, is the Euclidean distance between each pair of the 3D points. The configuration is nearly identical to the flattened version of the 3D surface of the CTHead. However, tangling still occurs in the configuration.

As shown in Figure 4-13, the residual errors mainly come from two issues: first, it is due to the difference between geometrical 3D structures and an impossibly perfect flattening process; second, "*the shortest-path metrics are generally non-Euclidean*" [126]. The result approximates a diagonal line, which would have been the geometrically impossible perfect flattening outcome.

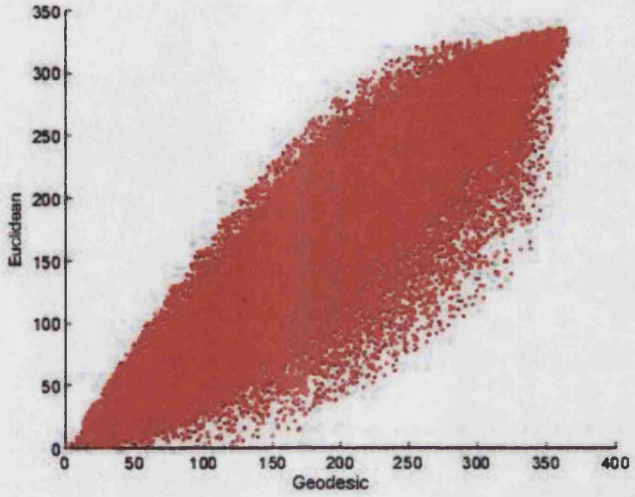


Figure 4-13: The shortest-path distance on the 3D surface (level sets) versus the Euclidean distance on the flatted 2D configuration. The data corresponds to the CTHead point cloud shown in Figure 4-12. The result approximates a diagonal line, which would have been the geometrically impossible perfect flattening outcome.

Flipping and tangling still exist

As shown in Figure 4-14, the configuration of classical MDS overlaps on the boundary nodes. Note that this is not an artifact. Metric-distance based classic MDS needs a third dimension to distinguish such local structures. Unfortunately, we deleted the third dimension information during the dimension-reduction process.

We notice that the tangling mainly comes from the boundary nodes. Therefore, an option is to adjust the length of shortest paths using *distance scaling* in the low dimensional cases [126]. However, distance scaling needs interactive minimisation, a process which is too time-consuming in real-time applications.

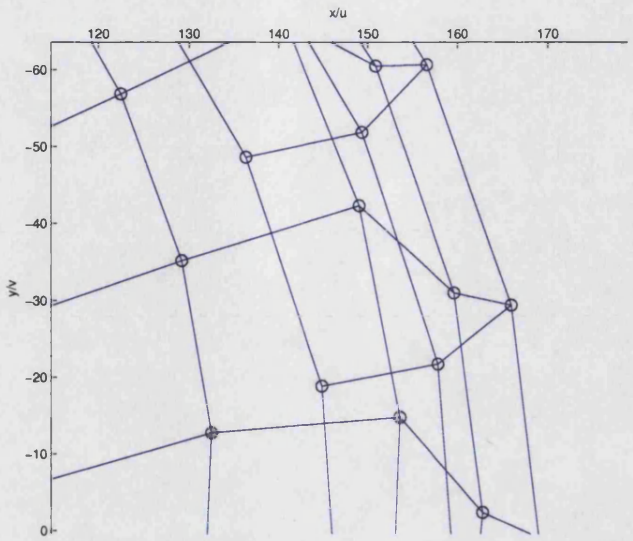


Figure 4-14: Close up of the area within the red box shown in Figure 4-12. Tangled areas still exist in the smoothed 2D configuration, using a shortest-path based proximity matrix.

Indexing positions on the flattened 2D plane

As shown in Figure 4-15, the evenly spaced sampling positions (circles) on an intermediate template (a spherical model is used here to render the intermediate template) were relocated onto the flattened areas (the red marks) of the surface of CTHead.

In other words, the indexing positions on the square intermediate template are now relocated to the indexing positions on the flattened surface of CTHead. The relationships between the indexing positions on an intermediate template and the indexing positions on the flattened configurations can be used to warp an intermediate template within the area of the flattened configuration. In addition, textures can be directly overlaid onto the flattened surface of the CTHead. An example of texturing volume objects is further given in the next subsection.

We used the minimal paths in the graph as the distances for MDS scaling. These distances are not Euclidean. Therefore, curvatures (not evenly distributed red sampling positions) in the configuration can also be introduced [126].

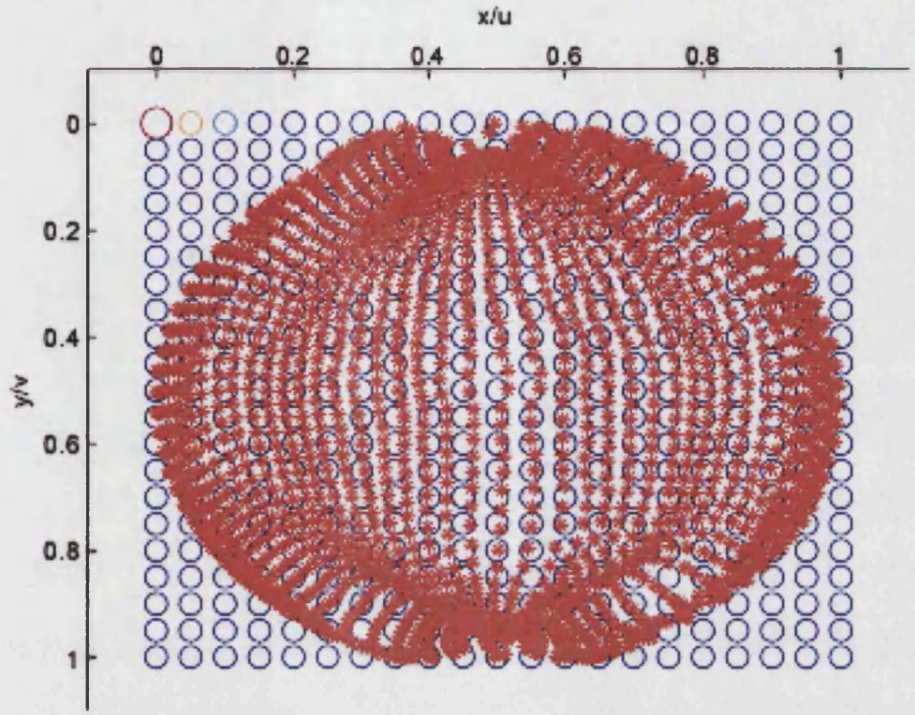


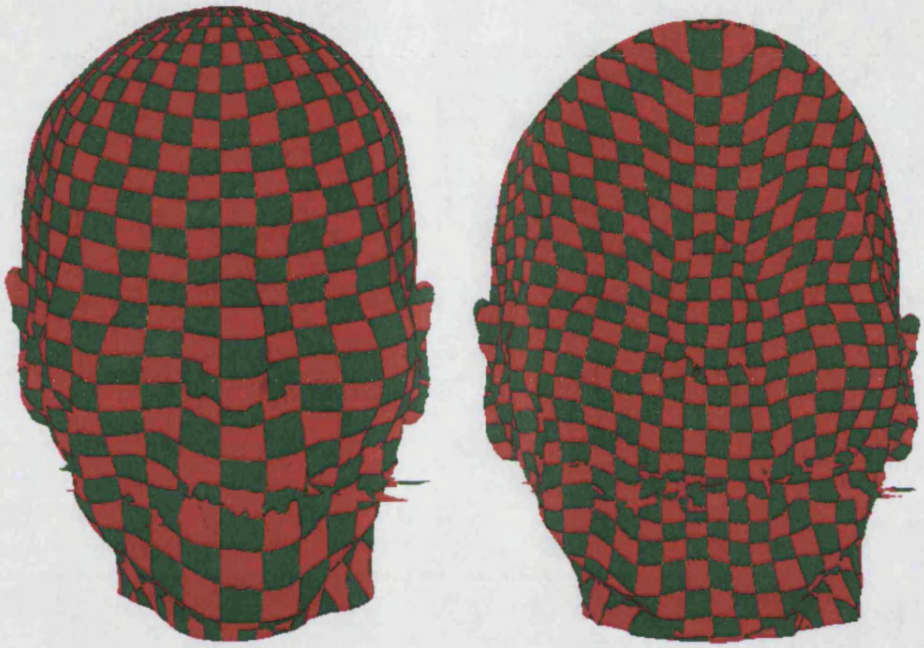
Figure 4-15: Sampling positions on the 2D plane of the flattened point cloud of the CTHead, using a shortest-path based proximity matrix.

Texturing a volume object

As shown in Figure 4-16, the CTHead data set was textured using a conventional spherical projective model (a) without and (b) with shortest-path MDS flattening control. The chessboard image was overlaid onto the flattened sampling positions shown in Figure 4-15. The circle sampling positions are evenly spaced. These sampling positions are used as origin positions to fire tracing rays in conventional spherical projective texture models. The texels gradually shrink towards the top of the head.

In contrast, the small red sampling positions were re-located on the flattened surface of the CTHead. The locations of these red sampling positions are warped within the areas of the flattened surface of the CTHead. The evenly spaced sampling positions in the conventional projective texture models now become the unevenly spaced sam-

pling positions located on the flattened surface. The chessboard texels can be overlaid on the area within the red sampling positions. The texels within the areas enclosed by the red sampling positions will be mapped onto the 3D surface of the CTHead. As shown in figure (b), texels can be directly overlaid onto the CTHead and the texels' shrinking (towards the top of the CTHead) can be eliminated.



(a) (b)

Figure 4-16: Texturing volume object using spherical projective model without flattening control (a) and with shortest-path MDS flattening control (b).

4.5.1 Computational Expenses

In our texture mapping pipeline, all the MDS operations are implemented as matrix operations in Matlab, as described by Zigelman et. in [38].

The flattened 2D positions are used as control points to reposition (u, v) coordinates on an intermediate template onto a flattened surface. For each position on intermediate template, we use lookup table to find its 4 control points of a quadrilateral. We use 26x26 control points to flatten the 3D surface of the CT-head object in this chapter.

Rendering Figure 4-16 take 14.17 seconds. The image size is 396x600 pixels and the running step length is set to 0.1. There are 4 point lights in the rendering scene. The size of the CT-head dataset is 180x113x237 voxels.

4.6 Conclusion

MDS is a class of techniques developed for the visualisation of high-dimensional data. These data are characterised by proximity (similarity or dissimilarity) values for all pairs of data elements. By interpreting the proximity as distances and constructing the flattened map, the so called MDS configuration can be used to explore the hidden structures of high-dimensional data sets.

There are several contributions we introduced in this chapter. First, by introducing neighbourhood relationships of 3D points, we have presented a new method for flattening 3D point clouds for texturing volume data sets. There is no need to construct mesh models as intermediate steps.

Second, we constructed a novel *plenoptic graph model* which is based on the plenoptic projective texture model to automatically estimate the shortest-path based proximity matrix in MDS.

Third, by using neighbourhood relationships, we can avoid the geodesic distance ambiguity problem. In particular, the *plenoptic graph model* is constructed from the DSOR representation, i.e, discrete, isolated sampling points. Therefore, the shortest-path estimation is robust to imaging noise and the resulting topological errors. Please note that distortions are inevitable, due to the well known "map maker problem" [38].

Our point cloud smoothing algorithm is not a mesh-based flattening technique. Though we just used a graph model to calculate the shortest-path proximity matrix for flattening points cloud using classic MDS, the basic representation of volume objects is still a point cloud.

Classic MDS analysis focuses on global properties within the proximity data. Therefore local properties might not be guaranteed with any reasonable configuration and interpretation. As we demonstrated in this chapter, if we would like to flatten the 3D point cloud of the CThead, tangling cannot be eliminated. So, in the next chapter, we will introduce a novel method to solve this problem using Laplaccian smoothing.

A comprehensive investigation of MDS in Volume Graphics might be worthwhile. The techniques offered in this chapter touch two applications of MDS, which include: (1) Dimensional reduction: Given high-dimensional data, compute a matrix of pairwise distances, and use classic scaling to find lower-dimensional configurations whose pairwise distances reflect the high-dimensional distances as well as possible. Classical scaling is identical to principal components when used for dimensional reduction. (2)

Graph layout: From the graph we derived a novel plenoptic surface based on shortest-path metrics, which is used to construct a proximity matrix to MDS for planar and spatial layout. Since shortest-path metrics are not strongly Euclidean, significant residual error will exist. Fully detailed discussions were given by Buja et al. in [125, 126].

Chapter 5

Linear-Weighted-Laplacian-Smoothing (LWLS) for Flattening Point Clouds

In order to eliminate the existence of tangling and twisting in MDS configurations, in this chapter we will introduce a Linear Weighted Laplacian Smoothing (LWLS) model to flatten 3D point clouds within volume objects. In addition, the LWLS method can preserve the continuity of the point cloud representation introduced in the previous chapter. The LWLS method integrates the benefit of the smoothing mechanism of Laplacian methods with the advantages of metric classic MDS methods.

We demonstrate a method that prevents the tangling of flattened points for texturing DSOR based volume objects.

In this chapter we will explain:

- A point cloud smoothing method combining the MDS method and the LWLS method. This will allow us to flatten the 3D points onto a 2D plane while guaranteeing that the smoothed 2D positions have no flipping and tangling.
- The proposed method contributes to boundary conditions of the LWLS method.
- The proposed method contributes to local multidimensional scaling analysis.

5.1 Introduction

By introducing a shortest-path based multi-dimensional scaling method, the plenoptic texture models were offered in the previous chapter. By flattening the 3D point cloud using the metric classic MDS, the methods benefit the projective texture models by overlaying texture images onto flattened surfaces of 3D volume objects in the 2D plane. Unfortunately, tangling of configurations still occur.

As we will demonstrate in this chapter: first, local differential representations can also be used to flatten a 3D point cloud within volume objects by using Laplacian framework; second, boundary conditions can be calculated using shortest-path based classic MDS, the technique described in Chapter 4.

5.2 Related Work

Many existing Laplacian framework representations were designed to support mesh editing, warping, and shape interpolation. Here we draw our inspiration mainly from detail preserving techniques, in particular the local connectivity.

From MDS to LWLS

Zigelman et al.'s MDS method is based on calculating geodesic distances on the *split* surface of a 3D object. The assumption is that geodesic distances running along different paths between two vertexes are equal to each other. Unfortunately, this assumption is not true for a 3D points cloud. This is ambiguity in calculating geodesic distances on an enclosed surface of a 3D volume object. In addition, we cannot guarantee the geodesic path always exists since what we know is only the point cloud rather than an explicit geometrical structure. Therefore, when we use the Euclidean distance between each pair of the 3D points for multi-dimensional-scaling, the flattened point cloud might be tangled.

We noted that Shontz and Vavasis presented a mesh warping algorithm for tetrahedral meshes using Linear Weighted Laplacian Smoothing (LWLS) [34]. Their methods not only guarantees a good quality warped mesh, with no tangled or flipped nodes, it also preserves the connectivity of the warped meshes.

With simplicity, their method first determines local weights for each interior node, then after applying an affine boundary transformation, the method solves linear equations to determine the final smoothed positions of interior nodes.

5.3 MDS-based LWLS Flattening

5.3.1 Mesh model based linear weighted Laplacian smoothing

Starting from the problem of preserving the connectivity of the neighbourhood of a point cloud of volume objects, we refer to the technique presented by Shontz and Vavasis [34]: Linear Weighted Laplacian Smoothing (LWLS). We are particularly interested in the theorem that gives the sufficient conditions for when a mesh can resist inversion using a specific transformation.

Given a continuous deformation of the boundary of a mesh model, LWLS can be used to track the movements of the interior nodes of the mesh model. However, we should keep in mind that LWLS cannot be guaranteed to work under all types of boundary transformations. The sufficient conditions of the LWLS smoothing methods depend on the status of the transformed boundary and can be described as follows:

Suppose the boundary nodes (under boundary transformation) have no tangling and flipping. Then if LWLS is used to reposition the interior nodes, the resulting mesh will have no tangling and flipping. Here we refer to the mesh tangling and flipping as the phenomenon that the lines of mesh models cross the other lines.

Linear Weighted Laplacian Smoothing can therefore be described as follows:
(1) First, generate a set of local weights, w_{ij} for each interior node (x_i, y_i) that represent the relative distances of the node to its neighbours, (x_j, y_j). The local weights can be calculated using the following equations:

$$max(\sum_{j \in N_i} log(w_{ij})) \mid w_{ij}, j \in N_j \tag{5.1}$$

$$\sum_{j \in N_j} w_{ij} = 1,\ w_{ij} > 0 \tag{5.2}$$

$$x_i = \sum_{j \in N_j} w_{ij} x_j \tag{5.3}$$

$$y_i = \sum_{j \in N_j} w_{ij} y_j \tag{5.4}$$

(2) Second, apply an affine transform to the boundary nodes. Using both these new boundary positions and the sets of weights w_{ij}, we can calculate the new positions of

internal nodes by solving the following linear equations [34]:

Let b and m represent the numbers of boundary and internal nodes. Define x_B and y_B to be vectors of length b which contain the new repositioned x and y coordinates of the boundary nodes. Define x_I and y_I be vectors of length m which contain the new x and y coordinates of the internal nodes to be repositioned. Then the Laplacian matrix, (for a weighted graph $G(V; E; w)$), can be defined as:

$$L(i,j) = \begin{cases} -w_{ij}, & if\ i \neq j \\ \sum_{k \in V} w_{ik} & if\ i = j \\ 0 & iff\ (i,j)\ not\ in\ E \end{cases} \tag{5.5}$$

Note that the boundary nodes are numbered last. By deleting the last b rows of the matrix L, we get $A = [A_I, A_B]$. Then A_I is an $m \times m$ matrix which contains coefficients corresponding to internal nodes and A_B is an $m \times b$ matrix which contains coefficients corresponding to boundary nodes. Using the Laplacian matrix A_I and A_B, and the new position matrix of repositioned boundary nodes $[x_B, y_B]$, the repositioned internal nodes $[x_I, y_I]$ can be calculated using the following linear equations:

$$A_I[x_I, y_I] = -A_B[x_B, y_B] \tag{5.6}$$

The LWLS method can be equal to the standard mesh parameterisation algorithm presented by Floater [131]: embedding a manifold 3D mesh with a boundary in the plane without foldovers. Floater's mesh parameterisation method, which is the so called convex combinations (barycentric coordinates), can be described as follows [132]:
(1) To each interior edge $e = (i, j)$ between internal nodes i and j, assign a positive weight w_{ij}, such that:

$$\sum_{j \in N_i} w_{ij} = 1 \tag{5.7}$$

where N_i is the set of vertices neighbouring the internal node i.
(2) To all other entries (i, j), assign $w_{ij} = 0$.
(3) Embed the boundary vertices in the plane such that they form a closed convex polygon.
(4) Solve the following equations for the coordinates of internal nodes:

$$(I - W)x = b_x \tag{5.8}$$

$$(I - W)y = b_y \tag{5.9}$$

where W is an $n \times n$ matrix containing w_{ij}, and b_x and b_y are vectors corresponding to the vertices *adjacent* to the boundary. The above equations show us the theorem that guarantees the *non-foldover* performance of LWLS transformations: Given a planar 3-connected graph with a boundary fixed to a convex shape in R^2, the positions of the interior vertices form a planar triangular mesh (i.e., none of the triangles overlap) if and only if each vertex position is some convex combination of its neighbour's positions.

Comparing equations (5.5), (5.6) with equations (5.8), (5.9), we have:

$$I - W = A_I = \begin{cases} -w_{ij}, & if\ i \neq j \\ 1 - 0 = \sum_{k \in V_i} w_{ik} = 1 & if\ i = j \\ 0 & iff\ (i,j)\ not\ in\ E \end{cases} \tag{5.10}$$

$$[b_x, b_y] = -A_B[x_B, y_B] \tag{5.11}$$

The above two equations give us a very important feature that, as long as the boundary is transformed into a 2D convex shape or convex combinations of its neighbours, then we can combine the connectivity preserving (no-foldover) of LWLS with the smoothing boundary of MDS. So what we have is the extension of LWLS: that is, how to generate a qualified 2D boundary of a 3D point cloud.

When we use MDS to smooth the point cloud, we actually solve the equations (4.4) and (4.5), which guarantee that the boundary points can be represented using eigenvector based combinations of all their neighbours. Note that all the boundary points on the projecting lines are actually projected onto a thin rectangle on the surface of the spherical model, a convex shape actually.

The above process can be divided into three steps: first, projecting 3D points onto the surface of a spherical model; second, smoothing the spherical 3D lattice grid using the shortest-path MDS method; third, the boundary condition is used in LWLS to relocate the internal nodes.

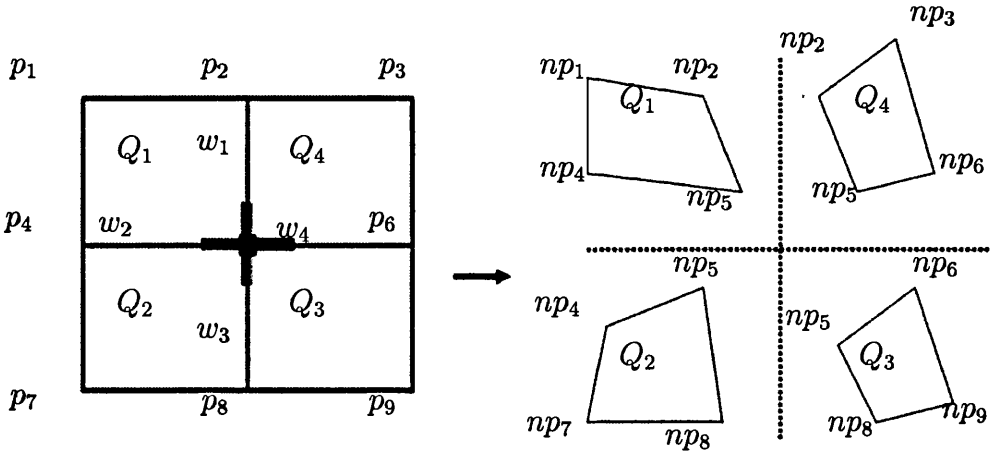
Based on the theory of mesh parameterisation with a virtual boundary [129] and the theory of the spherical parameterisation algorithms [132, 133], we can guarantee the existence of the one-to-one mapping between the standard evenly spaced sampling positions on spherical models and smoothed 2D configurations with MDS boundary

conditions. (The details of sufficient conditions are given in Appendix B). Therefore, we can prevent the flipping and tangling of internal nodes.

5.3.2 Principle of the algorithm: MDS-based smoothing weights

Given the point cloud shown in Figure 4-6, each 3D-point within the cloud can be indexed using the 2D lattice grid. An example is given in Figure 5-1(a). From the figure, we can see that a group of 3x3 3D points can be indexed by a 2D 3x3 lattice grid. Whatever the position of the 3D points, their connectively relationships can still be annotated using this 2D 3x3 lattice grid, which are the 3x3 neighbouring sampling positions on the plenoptic-based intermediate templates.

For each quadrilateral (represented by 2x2 neighbouring sampling positions), the benefit of our Euclidean distance based MDS method is that the MDS method can flatten 3D points with both minimal Euclidean differences and minimal differences of flattened areas. Therefore, for each of the adjacent 3x3 sampling positions shown in Figure 4-6(a), we can virtually connect a quadrilateral notation by connecting the adjacent positions, as shown in Figure 5-1(a). Each of the four quadrilaterals can then be flattened using our Euclidean based MDS (Flip a triangle to untangle the quadrilateral if it is tangled).



(a) 3x3 neighbouring sampling positions. (b) flattened quadrilaterals using *MDS*.

Figure 5-1: Flattened quadrilaterals using MDS: (a) The quadrilateral notation of a 3x3 sampling on the plenoptic surface shown in Figure 4-6(a); (b) The flattened four quadrilaterals.

As shown in Figure 5-1, the steps of calculating MDS-weighted smooth coefficients are described as follows:

(1) Given four quadrilaterals Q_1, Q_2, Q_3 and Q_4 and $p_i = (x_i, y_i)$, $i = 1, 2, ..., 9$, are the vertices of these four on the original intermediate template. Each vertex p_i is indexed with a 3D probed sampling point, $P_i(X_i, Y_i, Z_i)$.

(2) Flatten each quadrilateral using its four indexed 3D sampling points $P_i(X_i, Y_i, Z_i)$. We use a Euclidean distance based matrix M for multi dimensional scaling [38]:

$M = \{m_{ij} | i, j = 1, 2, 3, 4\}$
$m_{ij} = (X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2$
$B = -0.5 * J * M * J, where\ J is the centring matrix.$
$[Q, L] = eigs(B, 2, 'LM');$
$[newx_1, newx_2, newx_3, newx_4] = sqrt(L(1,1)). * Q(:, 1);$
$[newy_1, newy_2, newy_3, newy_4] = sqrt(L(2,2)). * Q(:, 2);$
$\{(newx_i, newy_i) | i = 1, 2, 3, 4\}$ are the flattened coordinates of the four vertexes.

(3) For each quadrilateral, repeat the above step, then $\{np_i(nx_i, ny_i), i = 1, 2, ..., 9\}$ are the new 16 smoothed positions of vertex p_i on the flattened plane. (Note that the positions of np_i will be different if they belong to different quadrilaterals.)

(4) The areas of the quadrilaterals are:

$S(Q_1) = \triangle(np_1, np_4, np_5) + \triangle(np_5, np_2, np_1)$
$S(Q_2) = \triangle(np_4, np_7, np_8) + \triangle(np_8, np_5, np_4)$
$S(Q_3) = \triangle(np_5, np_8, np_9) + \triangle(np_9, np_6, np_5)$
$S(Q_4) = \triangle(np_5, np_6, np_3) + \triangle(np_3, np_2, np_5)$
where $\triangle$ is the area of a triangle.

(5) Edge lengths (Euclidean distances) of quadrilaterals are represented by $EQ_m(np_i, np_j)$, where m is the index of the quadrilateral, $m = 1, 2, 3, 4$; np_i and np_j are vertexes on the quadrilateral, Q_m. Then the sum of edge lengths is:

$Sumedge = EQ_1(np_2, np_5) + EQ_1(np_4, np_5) + EQ_2(np_4, np_5) + EQ_2(np_8, np_5) + EQ_3(np_6, np_5) + EQ_3(np_8, np_5) + EQ_4(np_2, np_5) + EQ_4(np_6, np_5)$

(6) For an internal node p_5, its four MDS based smoothing coefficients are:

$$w_1 = (EQ_1(np_2, np_5) + EQ_4(np_2, np_5))/Sumedge$$
$$w_2 = (EQ_1(np_4, np_5) + EQ_2(np_4, np_5))/Sumedge$$
$$w_3 = (EQ_2(np_8, np_5) + EQ_3(np_8, np_5))/Sumedge$$
$$w_4 = (EQ_3(np_6, np_5) + EQ_4(np_6, np_5))/Sumedge$$

Sufficient conditions for preventing tangling are: convex combination coefficients $w_i > 0$, $i = 1, 2, 3, 4$, and, $w_1 + w_2 + w_3 + w_4 = 1$. As shown in Figure 5-1(a), w_1, w_2, w_3 and w_4 are further represented as colour bars, e.g, green (north), red (west), blue (south) and dark blue (east).

(7) For each sampling position on the plenoptic surfaces, repeat steps (1)-(6), to calculate their own smoothing coefficients.

(8) Calculate the total *area* as the sum of all the flattened quadrilateral areas. Then the length of the edge of the square is:

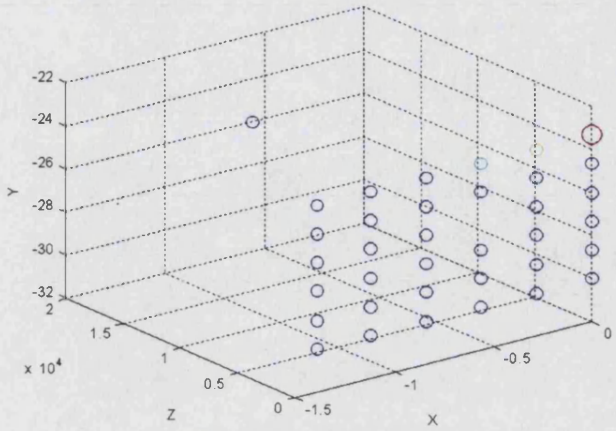
$$Edgelength = Square\,root(area).$$

(9) Finally place 2D *boundary* points evenly-spaced around a square of this size. All these re-positioned boundary points will be used as boundary conditions, $[x_B, y_B]$, in the Linear Weighted Laplacian Smoothing algorithm.

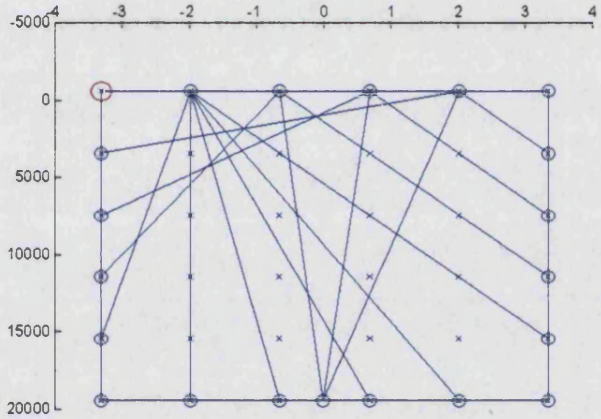
5.4 Smoothing a Point Cloud Using MDS-Weighted LWLS

In Figure 5-2 we give an example of calculated MDS weights using a synthetic 3D point cloud.

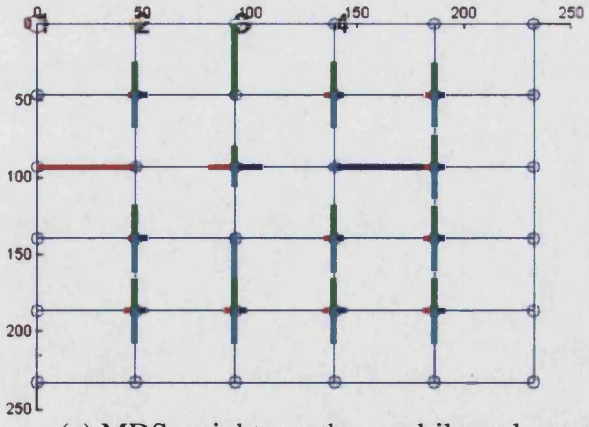
Figure 5-2(a) shows a synthetic 3D planar dataset, except that the center point is lifted to test the algorithm. Figure 5-2(b) shows the flattened point cloud, using Euclidean distance based MDS methods, is tangled. Figure 5-2(c) shows the calculated MDS weights.



(a) 3D synthetic dataset.



(b) Tangled point cloud.



(c) MDS weights on the quadrilateral notation.

Figure 5-2: MDS weights: (a) The 3D synthetic dataset. (b) Point cloud is smoothed using Euclidean distance based MDS(anchored boundary). (c) The strength and direction of MDS weights are represented by the length and direction of colour bars.

For each internal node p_i, (x_i, y_i) are its x and y coordinates on the original intermediate template. Driven by the MDS-based weights, w_{i1}, w_{i2}, w_{i3} and w_{i4}, p_i will be re-positioned within the flattened square using the LWLS method [34]. The flattened point colud is shown in Figure 5-3(a). Here i is the indexing number of all the internal nodes and w_{ij} represents the weight of the ith node given by the jth. Then the Laplacian matrix, L, can be defined as:

$$L = \begin{cases} -w_{ij}, & if\ node\ j\ is\ neighbouring\ node\ i \\ 1 & if\ i = j \\ 0 & else \end{cases} \tag{5.12}$$

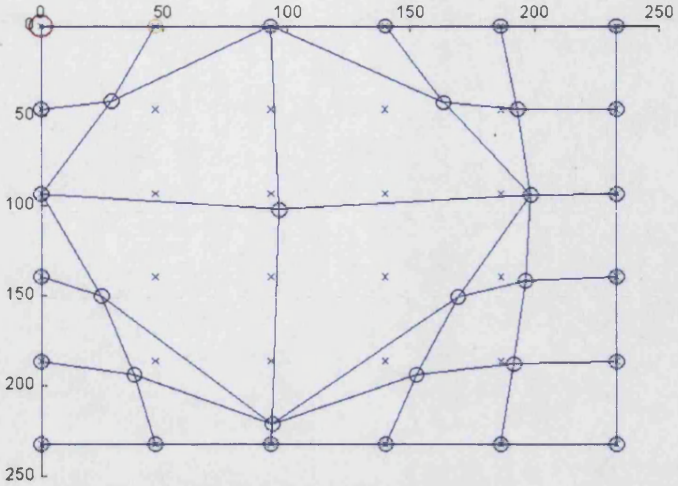
$$w_{ij} = \begin{cases} -w_{i1}, & if\ node\ j\ is\ the\ north\ neighbour \\ -w_{i2}, & if\ node\ j\ is\ the\ west\ neighbour \\ -w_{i3}, & if\ node\ j\ is\ the\ south\ neighbour \\ -w_{i4}, & if\ node\ j\ is\ the\ ease\ neighbour \end{cases} \tag{5.13}$$

Here we also number boundary nodes last. Following Shontz's annotations [34], matrix $A = [AI, AB]$ by deleting last boundary rows in matrix L. $[x_B, y_B]$ are the flattened coordinates of boundary nodes estimated in the above step (9). Then the flattened coordinates of the internal nodes are:

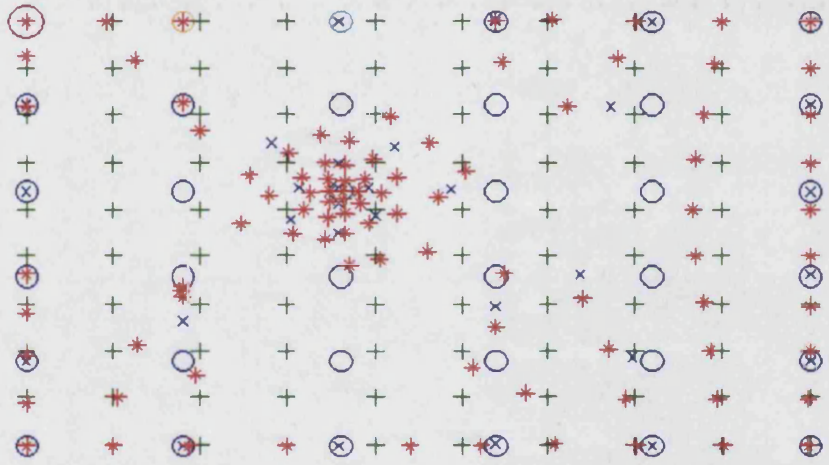
$$[x_I, y_I] = -AI./AB[x_B, y_B] \tag{5.14}$$

The MDS weights ensure the sampling positions on the intermediate template are now re-positioned over the flattened 3D surface. The boundary condition preserves the flattened area of the 3D surfaces and thus ensures sufficient texels for rendering high quality close ups. Here, the boundary is convex. As we will demonstrate later, a real boundary of a flattened 3D surface can also be used as the boundary condition in LWLS.

The beauty of this flattening control is, if we overlay evenly spaced sampling positions onto the flattened surface, then they will be indexed to un-even sampling positions on the original intermediate template.



(a) Flattened quadrilateral notation.



(b) Multiresolution positions on the intermediate template.

Figure 5-3: Smoothing point clouds using MDS weighted LWLS: Warped sampling positions on intermediate template. (a): "x": 6x6 evenly spaced sampling positions on the original intermediate template; "o": 6x6 smoothed sampling positions on the flattened intermediate template. (b): "*": 10x10 warped sampling positions on the original intermediate template; "+": 10x10 evenly spaced sampling positions on the flattened intermediate template; "x": 6x6 smoothed sampling positions on the flattened intermediate template; "o": 6x6 evenly spaced sampling positions on the flattened intermediate template.

An important feature that we find is that these un-even sampling positions on the original intermediate template actually represent the high-resolution samplings of the

details of different areas on the original intermediate template. These un-even sampling positions within different areas (close ups) can be defined as *multi-resolution representations of the intermediate template.*

Note that in Figure 5-3(a), "o" represents the re-positioned sampling positions on the flattened intermediate template; "x" represents the evenly spaced sampling positions on the original intermediate template.

In Figure 5-3(b), we first overlayed evenly spaced sampling positions onto the *smoothed* quadrilateral annotation. "+" represents the 10x10 evenly spaced sampling position. Then, each sampling position "+" will be re-positioned to position "*" according to the inverse smoothed/flattened control. Then "*" represents the re-positioned 10x10 sampling positions on the original intermediate template; "x" represents the re-positioned 6x6 sampling positions on the original intermediate template, "o" represents the evenly spaced 6x6 sampling positions on the flattened intermediate template. Finally, if we render the intermediate template using the "*" sampling positions shown in Figure 5-3(b), then we get a flattened intermediate template.

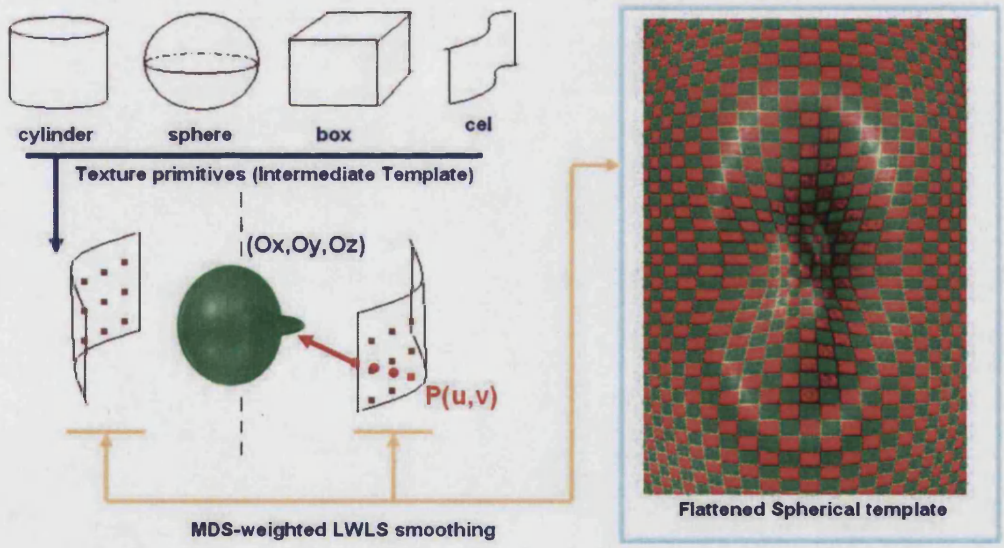


Figure 5-4: Texturing volume objects using a flattened intermediate template. The flattened spherical template is an intermediate using the spherical model.

In summary, we overlay the evenly spaced sampling position "+" onto the smoothed (flattened) surface. Then we re-position these evenly spaced sampling positions onto

the non-smoothed annotations. We are adjusting the sampling positions on the plenoptic surface. These sampling positions are used as starting point for firing tracing rays.

Figure 5-4 gives the texture pipeline: first, construct the flattened intermediate template; second, directly overlay an image onto the flattened intermediate template; third, during texture mapping, the indexing position on the intermediate surface was re-positioned onto the flattened position, at which the texture image is overlaid. A flattened spherical intermediate template, the original green volume object and different texture projection primitives implemented in our system are shown in Figure 5-4.

Figure 5-5 gives the textured volume dataset using standard planar mapping [40] and flattened planar mapping. This time we use planar projection for texturing volume objects. The results show reduced shear effect.

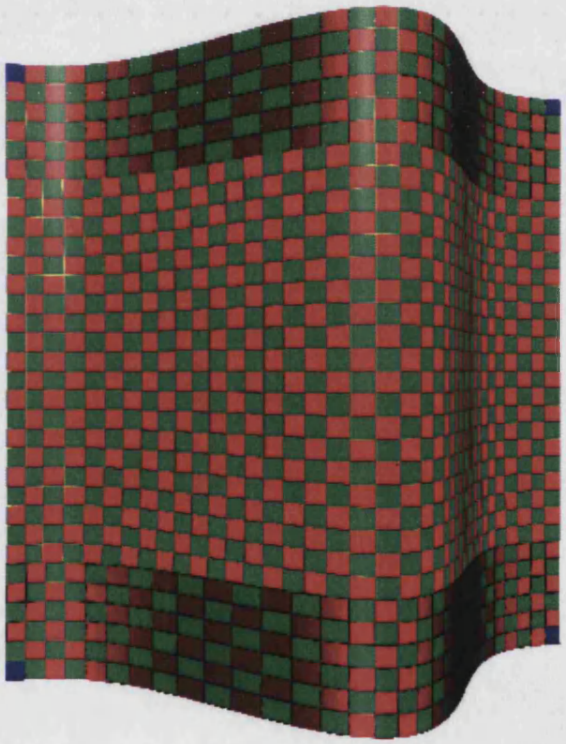


Figure 5-5: Volume object is textured using MDS-weighted LWLS smoothing method (highlighted textures) and standard projective texture models discussed in [40] (shaded textures). The results show that the shear effect in the highlighted areas is much less than those in the shaded areas.

5.5 MDS Boundary Conditions in LWLS

As described in previous sections, in order to preserve the connectivity of the vertices during mesh smoothing, boundary positions must be smoothed into a convex 2D shape [129]. We demonstrated the smoothing effect using a square boundary in the previous sections. Even though the tangling and twisting of points are eliminated, the distortions around the boundary areas are still too high compared to the distortions of internal nodes. In other words, if we would like to overlay a texture photo onto the smoothed intermediate template, we still need to pay attention to the distortions around the boundary areas to make sure textures can be pinned onto the associated key features within.

Here, we present a combination of MDS and LWLS that resolves this high distortion problem by substituting the boundary shape in LWLS with MDS smoothed boundary positions. In the ninth step described in subsection 5.3.2, the 2D boundary points are evenly placed around a square. All these square based boundary positions are used as boundary conditions in LWLS in the previous sections.

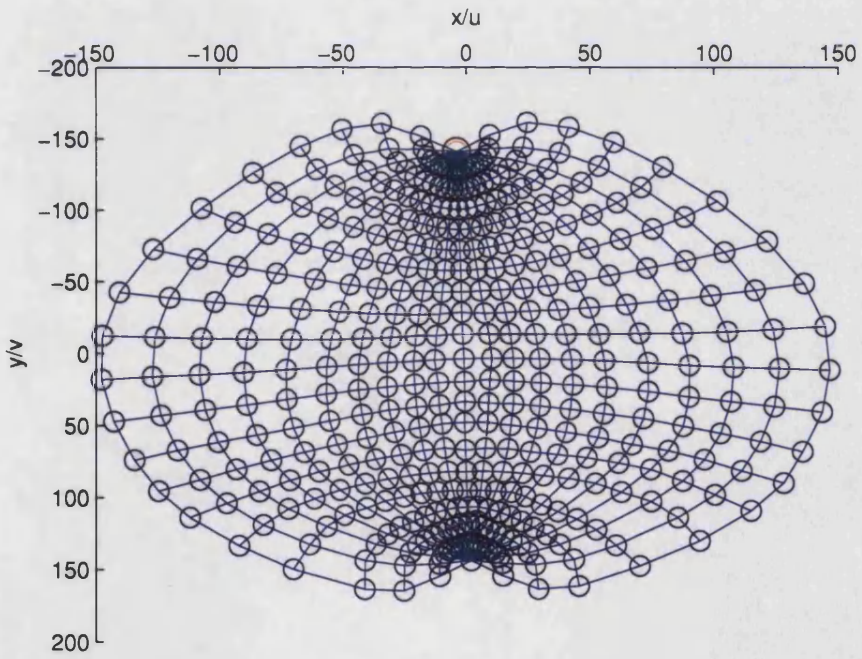


Figure 5-6: Flattened 3D point cloud using MDS-boundary based LWLS smoothing.

In order to use the MDS boundary condition, we calculate MDS smoothing for all the points. Then we use the MDS smoothed boundary positions, $[x'_B, y'_B]$, in equation 5.14. The MDS-boundary based LWLS smoothed positions, $[x'_I, y'_I]$, can be calculated using the following equation:

$$[x'_I, y'_I] = -AI./AB[x'_B, y'_B] \tag{5.15}$$

The smoothed point cloud using this method are shown in Figure 5-6. The internal nodes are smoothed within the real boundary, without any flipping and tangling.

5.6 Experimental Results

We show here some more results, containing various aspects of the complete methods.

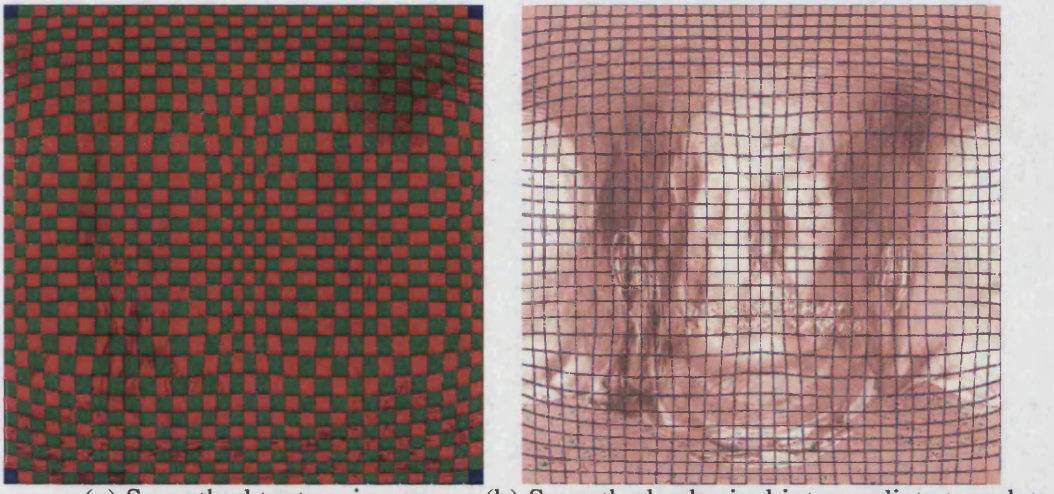
(a) Texture image (b) Spherical intermediate template

Figure 5-7: (a): Texture image for annotating CT head. (b): Standard spherical intermediate template.

Figure 5-7(a) is a texture image used for annotating a volume CT head. Figure 5-7(b) is a spherical intermediate template of this CT head. Since there is no colour information here, in order to see clearly the effect of surface flattening, we first texture the CT head using Figure 5-7(a), then flatten the spherical intermediate template using our MDS weighted LWLS method.

As shown in Figure 5-8(a), the evenly spaced texture blocks were flattened according to the shape of the 3D surface of the CT head. In Figure 5-8(b), we overlay the

edges of the texture image shown in Figure 5-7(a) onto the standard spherical template shown in Figure 5-7(b). The flattening effect is obvious; the deeper the 3D surface, the more texels will be applied to it.



(a) Smoothed texture image (b) Smoothed spherical intermediate template

Figure 5-8: (a) Flattened spherical intermediate template of the CT head using the MDS-weighted LWLS smoothing method. (b) Overlayed edges of the left image onto the MDS-LWLS smoothed spherical template (with square boundary condition). Note that the deformed quadrilaterals reflect the flattened surface on the intermediate template.

Two further examples are given in Figure 5-9. The shaded texture on the CT head was textured using our previously discussed standard texture mapping system [39, 40]. The highlighted textures were textured onto the CThead using MDS-weighted LWLS methods.

Note that more texels are embedded in the highlighted texture areas, which demonstrates the reduction of the shear effect. The larger the area of the surface, the more texels can be embedded in the surface. The size of the different areas on the smoothed intermediate template of the surface depends on the geometrical structure of the different areas on the 3D surface.

We can use either DSR or DVR to render the image. It is worth pointing out that here we are still using the spherically projective model to do texture mapping (with and without MDS-weighted LWLS smoothing control). A novel warped intermediate template which can reduce texture shrinks will be introduced in the next subsection.

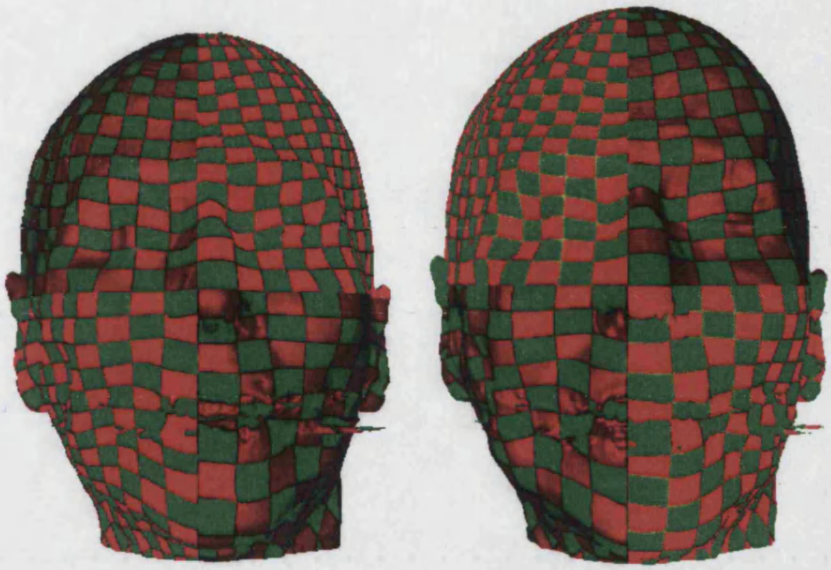


Figure 5-9: Textured volume object using the MDS-weighted LWLS smoothing method (highlighted textures) and our previously discussed semantic spherical texture model (shaded textures) [40]. Note that more texels are embedded in the smoothed areas (highlighted textures), which demonstrates the reduction of the shear effect. The larger the area of the surface, the more texels can be embedded in the surface.

5.6.1 Warped intermediate template

By using the MDS-LWLS flattening, we warp the standard intermediate template from a rectilinear shape to a flattened surface shape, which is based on the boundary based LWLS smoothing algorithm.

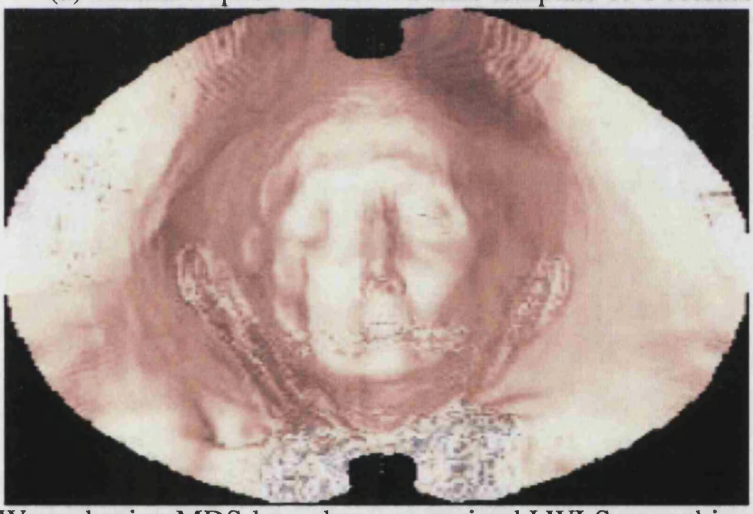
Figure 5-10(a) is the standard square intermediate template. If the texture image (chessboard) is overlaid onto figure (a), then the texture will shrink onto the top of the head, as shown in Figure 5-9.

In comparison, if we warp the intermediate template using the warping control shown in Figure 5-6, then the intermediate template is smoothed into a flattened surface of the CT-head. Then if the texture image is overlaid onto figure (b), which is the flattened surface of the CTHead, the texels will be nearly uniformly textured on the surface.

As shown in Figure 5-11, figure (a) is the textured CTHead using a standard spherical intermediate; figure (b) is the textured CTHead using the warped intermediate

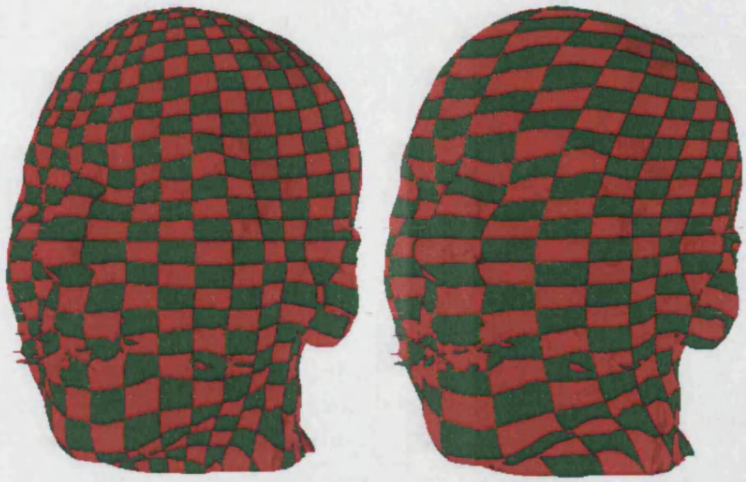
template. In figure (b) the texture image is overlaid onto the warped intermediate template, Figure 5-10(b), which is the flattened surface of the CTHead.

(a) Standard spherical intermediate template of CTHead.



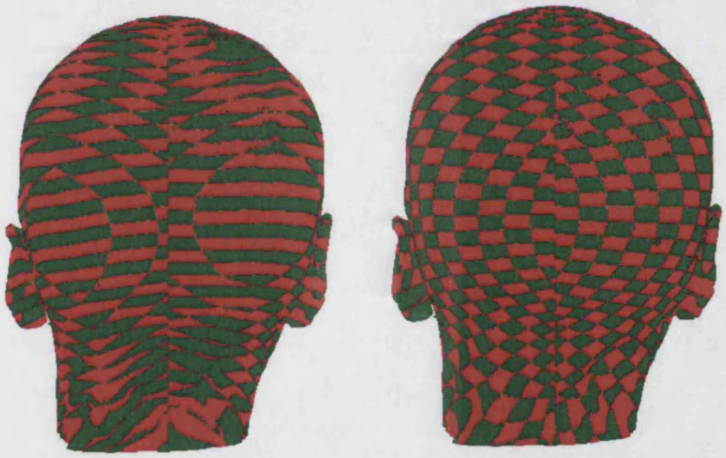
(b) Warped using MDS-boundary constrained LWLS smoothing control.

Figure 5-10: Warping the intermediate template: (a) The standard spherical intermediate template (square) of the CT Head. (b) A warped spherical intermediate template of (a) using the MDS-boundary constrained LWLS smoothing method. The texture image (chessboard) can be directly overlayed onto the flattened template (b).



(a) (b)

Figure 5-11: Textured CTHead: (a) The CT Head is textured using the standard spherical intermediate template. (b) The CT Head is textured using the warped spherical intermediate template. In figure (b), the intermediate template is warped using the MDS-boundary constrained LWLS smoothing method.



(a) (b)

Figure 5-12: Textured volume object using the standard classic metric MDS flattening technique (a) and the MDS-boundary constrained LWLS smoothing method (b). Image (a) is rendered using the flattened surface shown in Figure 4.13. The textures on the boundary areas are flipped and tangled. Image (b) is rendered using the MDS-boundary-LWLS techniques (the flattened surface shown in Figure 5.6), thus there is no texture tangling and flipping.

5.6.2 Computational Expenses

In our texture mapping pipeline, all the LWLS operations are implemented as matrix operations in Matlab, as described by Shontz and Vavasis in [34]

The flattened 2D positions are used as control points to reposition (u, v) coordinates on an intermediate template onto a flattened surface. For each position on intermediate template, we use lookup table to find its 4 control points of a quadrilateral. We use 26x26 control points to flatten the 3D surface of the CT-head object in this chapter.

Rendering Figure 4-16 take 14.17 seconds. The image size is 396x600 pixels and the running step length is set to 0.1. There are 4 point lights in the rendering scene. The size of the CT-head dataset is 180x113x237 voxels.

5.7 Conclusion

Starting from solving the problem of tangling and flipping of the point cloud, we used the MDS-boundary condition to embed the internal point set into the nearly-identical boundary of the flattened 3D surface. MDS-weights were used as smoothing weights in the LWLS framework.

We demonstrated that the MDS-boundary condition is a practical method to generate a nearly-identical smoothed boundary of a 3D surface. In particular, such a 3D surface can be a generically enclosed model without any explicit boundary definition. It is worth pointing out that we do not need to generate any 3D virtual boundary for the 3D enclosed surface [129], and in addition, we do not need to split the enclosed 3D surface into patches to control the quality of texture mapping. More texels can be embedded into large areas in the smoothed surface. Therefore the shear effect can be reduced when texturing or annotating volume objects.

Our point cloud smoothing algorithms cannot be considered as a mesh-based flattening technique, though geodesic MDS methods [38] or LWLS methods [34] are presented as mesh-based smoothing techniques. In our implementation, we never build an explicit connectivity between adjacent sampling positions. The basic representation of a volume object is still a point cloud.

Mesh-model based surface smoothing techniques can be directly migrated into our volumetric texture mapping pipeline. The beauty of the quadrilateral lattice grid representation is that it can be used to migrate a wide range of mesh smoothing algo-

rithms [130]. As we demonstrated, shape-preserving algorithms [131] are combined with a MDS boundary condition in LWLS framework. The quadrilateral lattice grid is actually the sampling positions on the spherical model; therefore, our MDS-boundary based LWLS method can be a spherical based parameterisation method for 3D point clouds. Our method can be thought of as an extension that is highly relevant to the spherical parameterisation for 3D meshes [132].

We believe that, using a variety of semantic constraints, for instance, space constraints and logical constraints, the intermediate template based volume texture mapping pipeline is be a practical solution to applications of volume visualisation such as medical training and studio production.

Chapter 6

Projective Masking Fields

Spatial constraints, geometrical constraints, topological constraints, and logical constraints may be used to split or segment volume data sets. These constraints play critical roles in annotating, texture mapping and visualising volume objects, in particular in applications such as medical training, studio-production, and entertainment. In this chapter, we present a novel space splitting method, using projective (planar, cubic, cylindrical, and spherical) models, to split iso-surfaces which are self-occluded. Our technique is also a solution to the texture penetration problem of the pseudo-solid texture model, which we discussed in the previous chapters.

We present a solution to the texture self-occlusion problem and the texture penetration problem, by splitting the 3D space into different labelled layers. Using labelled 3D masking fields, self-occlusion and texture penetration can be effectively controlled using volume rendering techniques, such as DVR and DSR. We do not build masking fields as a pre-constructed scalar field before volume rendering. In contrast to the techniques of constructing 2.5D pseudo-solid texture models, we dynamically construct masking fields during volume rendering.

In this chapter we will explain:

- Splitting and labelling 3D volume space using projective (planar, plenoptic) models.
- Constructing masking fields of 3D volume space during volume rendering, rather than constructing masking fields as a pre-processing step.
- Splitting self-occluded iso-surfaces using the new masking constraints.

6.1 Introduction: Texture Self-Occlusions and Penetration

Texture penetration is a critical issue in the previously discussed pseudo-solid texture models. As shown in Figure 3-3(d), although we can use direct surface rendering to render the iso-surface, textures can still penetrate from the exterior layer (skin) of iso-surfaces into the interior layer of the *maxillary sinus* (the same level sets of the skin).

As shown in Figure 6-1, given an iso-surface with multi-layered structures, the projective colour indexing mechanism cannot tell at which position the iso-surface should be textured and at which position the iso-surface should not be textured. We will offer a solution, field masks (semantic layers), to solve this problem.

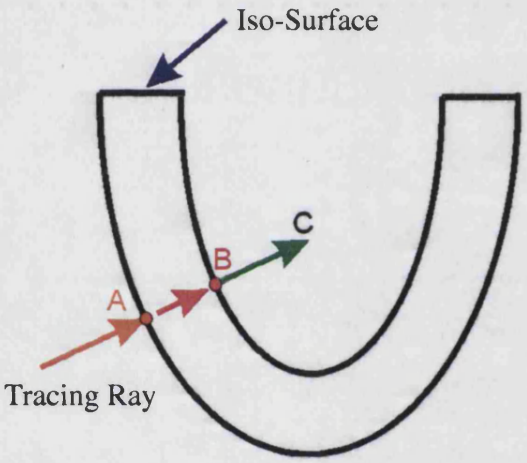


Figure 6-1: Iso-surface: Self-occlusion. Given an iso-surface, A and B are two positions at which the tracing ray passes through. If we fire a tracing ray towards C, position B will be occluded by position A.

The direct surface rendering technique (DSR) can pick up the iso-value at a specific 3D position. So if we wish to have different texture at position A and position B, then additional space constraints must be provided.

Suppose a tracing ray passes through positions A and B which have the same iso-value. Position A will occlude B's texture information, since position A will be detected first. Therefore, by also labelling the detecting order of the different positions on the iso-surface, we can effectively split the 3D space to texture position A and position B independently.

6.2 Related Work

Splitting 3D space is a basic research topic in the volume community. However, to our knowledge, using projective models (plenoptic or planar based) to split volumetric space has not been reported anywhere. This chapter offers the technique to split volumetric space using a projective model based direct surface rendering technique.

We will first discuss the traditional z-buffering algorithm. Then we will discuss the self-occlusion problem of conventional mesh models. Following this, we will discuss the volumetric splitting model.

Z-buffering algorithm

The famous z-buffer algorithm was originally developed for rendering the visible-surface [134]. For each pixel in rendering image, a z-value (depth information) is stored into a buffer z. The z-buffer is initialised to infinity, representing the z-value at the back clipping plane. The smallest value that can be stored in the z-buffer represents the depth information of the front clipping plane. If a polygon point is no farther from the viewer then the depth-information is saved in the z-buffer. The z-buffer value is updated by the new scanned polygon point.

Note that, first, the consequence of the z-buffering algorithm is that the 3D space is split into two semantic areas: visible or invisible, from the rendering (viewing) direction; such semantic information does not exist on the polygon model itself. Second, the visibility information is dynamically generated during the process of rendering; Third, the occlusion information is highly specific to the viewing direction. This means that the occlusion is dynamic, which is relevant to rendering configurations.

As we will explain later, the above four properties of the z-buffer algorithm will be implemented in our volumetric splitting model.

Self-occlusion

Splitting space in volume graphics was addressed by Lischinski and Rappoport [135]. Their methods are particularly of interest to us. First, as shown in Figure 6-2, the layer-depth information was saved at each pixel, ordered along the projective direction. In particular, the samples e, f, g and h, are located on the same object. These four samples could be self-occluded if being viewed from pixel 9. Second, Lischinski and Rappoport's methods consist of image-based rendering, light field rendering and volume graphics. Their layered-depth cube representations consolidate different scene representations into a common framework. In other words, the layered depth

cube method combines view-independent scene information and view dependent appearance information.

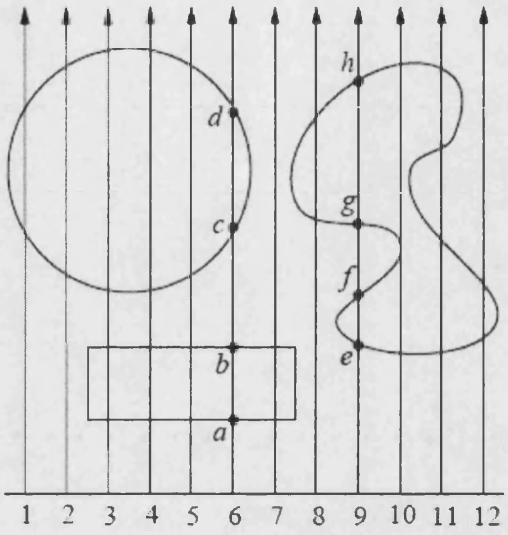


Figure 6-2: A parallel layer depth image of a 2D scene. Pixel 6 stores scene samples a, b, c, d, pixel 9 stores samples e, f, g, h. Figure from [135].

View-independent scene information, such as geometry and diffuse shading, is represented using three orthogonal high-resolution layer depth images. The view dependent scene information is stored as a separate, larger collection of low-resolution layer depth images. The rendering algorithms combine these two components using two steps: first, 3D warping of the layer depth images and filling holes, using the ray tracing technique. This stage results in the image which represents the geometry of the scene as seen from the new viewpoint. Second, this image can be shaded using a local shading model and reflection can be calculated using environmental mapping techniques. To our interest, Lischinski and Rappoport's method does present a novel combination of different techniques, in particular in adopting volume graphics and splitting volume space using view-independent projection models.

Volumetric splitting model

In additional to Islam et al.'s splitting constraints [14], in this chapter we will present a plenoptic / planar based projective splitting model. We will present a novel combination of volume rendering techniques (DSR and DVR), plenoptic modelling techniques, field functions and spatial transfer functions.

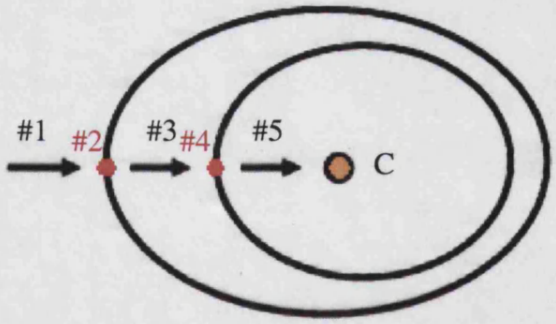
6.3 Volume Rendering: Semantic Constraints and Semantic Field

6.3.1 Volume rendering: traversing 3D space

By firing tracing rays, 3D space can be traversed at fixed step-length. The scalar field values at the termination positions of tracing rays can be calculated. Conventional volume rendering engines use these properties to render the images.

Here we use this mechanism in a novel way, i.e., constructing field masks by sweeping the 3D volumetric space using tracing rays. We refer to the tracing rays which are used to label the field masks as the *masking rays*. Here, masking rays are fired from the positions on a projective masking model (planer, box, cylinder and sphere). Each masking ray is associated with a variable number of indices. A *masking field* can therefore be defined as the positions of a set of 3D point clouds which have the same indexing number.

The iso-surface in Figure 6-3 has multiple layers that have the same iso-value. We would like to split the 3D space into differently labelled layers (masking fields) such as layers 1, 3 and 5. Note that 1, 3 and 5 are labelling numbers of 3D space, whereas, 2 and 4 are labelling numbers of iso-surfaces.



C: centre of projection

Figure 6-3: Semantic constraints: projective masking fields.

Figure 6-3 shows a masking ray running towards the 3D position *C*. Masking rays split the 3D space into different layers (masking fields). Note that the masking field is not a static model of the volume object. If we use different projective models to fire masking rays, then masking fields will be different accordingly. This is the property

we want to use, that is, we can select different projective models to construct different masking fields, according to the 3D structures of iso-surfaces. Different masking fields can be used together to construct more advanced logical constraints.

In Chapter 3, we introduced projective texture based on plenoptic projection. Plenoptic projection not only suits a variety of geometrical structures for environment texture mapping, but can also sweep 3D space along projective directions. These two properties are the characteristics we want to use in our algorithm, i.e., flexibility of representing geometrical structures of iso-surfaces and efficiency of sweeping 3D space. Therefore, we use plenoptic models to fire masking tracing rays. First, we can select an appropriate plenoptic model to construct the field masks, according to the geometrical structure of the iso-surfaces. Second, projective texture models and field masks can have different plenoptic models. These two properties provide us extreme flexibility in volume rendering.

6.3.2 Tracing a ray: labelling constraints

Plenoptic models, projective indexing and volume rendering are consolidated into our volume rendering engine. In other words, volume rendering, scalar-field based modelling, and the previously discussed field function control and spatial transfer function control can be integrated.

As discussed in the previous chapters, scalar fields are properties of volume objects. The scalar fields could be intensity values of the original scan, the optical properties of the objects, subsequently-constructed geometrical models, velocity / flow fields, and so on. Therefore, if we sweep scalar fields using tracing rays, we can sweep the whole 3D volumetric space and also pick up the necessary scalar values.

As shown in Figure 6-4, masking fields can be calculated during volume rendering, so there is no need to calculate a static masking field in advance. The rendering pipeline is now composed of the following steps:
(1) In order to render pixel $P(x, y)$ of the image, we fire a ray towards the volume object.
(2) The ray terminates at the 3D position (P_x, P_y, P_z) on the iso-surface. In order to find the masking field value on this 3D position, we need to fire a masking ray, from position $P(u, v)$.

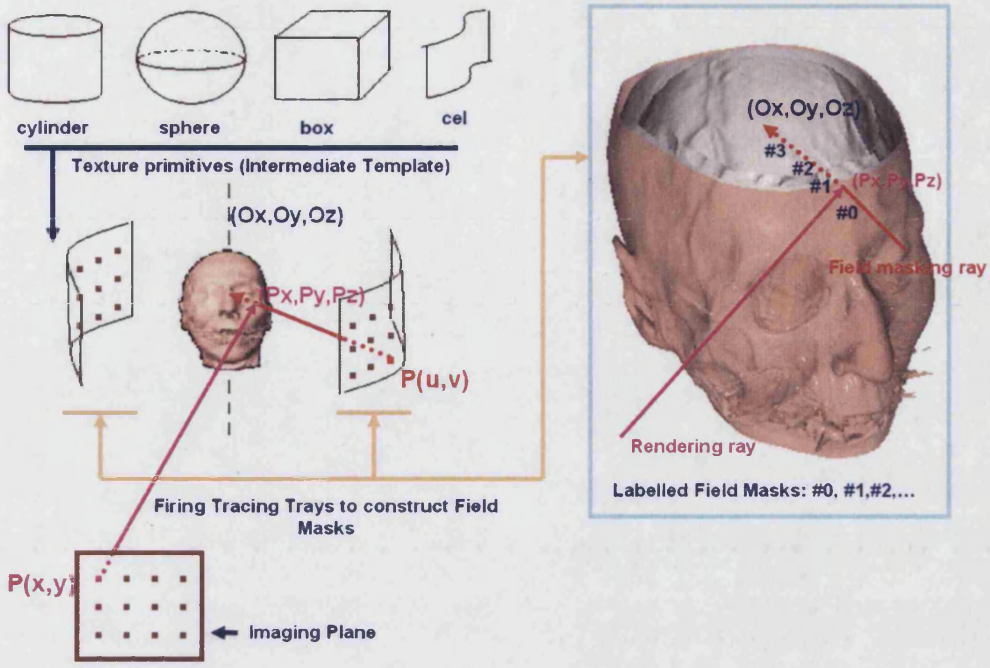


Figure 6-4: Volume rendering: projective masking fields and marking rays.

(3) Position $P(u, v)$ is the intersection between the masking ray and the plenoptic model, such as a sphere, cube or box.
(4) The masking ray starts from $P(u, v)$ and runs toward the centre of the plenoptic model, (O_x, O_y, O_z).
(5) If the masking ray meets an iso-surface, then its masking-field variable will be increased, otherwise, the masking-field variable is unchanged.
(6) If the rendering tracing ray runs through position (P_x, P_y, P_z), the masking-field value at this position will be equal to the current masking-field value of the masking ray.

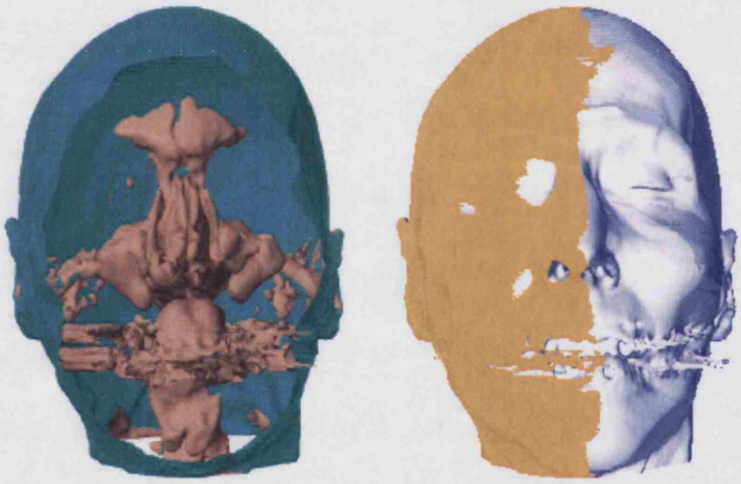
From Figure 6-4, we can see that the masking fields at different iso-surface positions on the same masking ray are labelled as $\#0$, $\#1$, $\#2$, $\#3$ accordingly.

Different plenoptic models can be used in projective texture and masking fields. For instance, in order to texture map a volume object, we can use a spherical model for texture indexing, as we described in Chapter 3. Meanwhile, we can use planar projection to calculate masking fields.

The benefits of this flexibility are shown by Figure 6-5. We split the CTHead into exterior and interior layers (a) using a spherical masking field model, and into the left

and right layers (b) using a planar masking field model (projecting masking rays from left to right).

Cylindrical, spherical, cubic or planar based geometrical model can be used to calculate field masks. The selection criteria are similar to those for plenoptic models for environmental mapping. In addition, projective texture, which was introduced in Chapter 3, can be spatially controlled using masking fields. As shown in Figure 6-5(a), the exterior masking field (green layer) and the internal masking field (light pink) can be textured independently. As shown in Figure 6-5(b), the exterior layer can be further split into left and right masking fields, therefore, the left side of the face and the right side of the face can be textured independently. Additional examples will be given in the following sections.



(a) (b)

Figure 6-5: Texturing volume objects using field masks. (a) The iso-surface of the CTHead is split into exterior (skin) and interior (internal tissues such as tongue and ear channels) layers, using a spherical masking field. (b) The iso-surface of the CTHead is split into left and right parts, using a planar masking field. Different portions of the iso-surfaces can therefore be textured or coloured independently.

So spatial constraints, iso-values and masking labels can be used together to texture map a volume data set more effectively. Note that the above images were rendered using a single iso-value. In other words, the iso-surface of the skin, the nose, the

tongue are the same iso-surface.

Therefore, we offer a powerful tool to split the iso-surface flexibly, without the need for additional geometrical information. In particular, the masking field can be treated as an additional scalar field. The beauty of scalar field based models, which we discussed in Chapter 2, enables us to generate novel properties flexibly and efficiently.

6.4 Semantic Volume Splitting

We present a new spatial constraint, masking field, to effectively split the volumetric space. The masking field could thus be used as a novel semantic constraint to the volume splitting model which is described in Chapter 2.

(a) (b)

Figure 6-6: Textured CTHead using projective marking field: (a) The iso-surface of the skull is split into exterior and interior layers. Using projective masking fields, these two layers can be textured independently. (b) The iso-surface of the skin of the CTHead is split into exterior and interior layers. The skin of the CTHead and the interior layers (tongue, ear channels, soft tissues) can be textured independently.

Two more examples are given in Figure 6-6. Given the iso-surface, i.e., the skin or the skull of the CTHead, the exterior and the interior can be split using spherical

masking field. The different layers can be textured independently since they have different masking field indices.

In Figure 6-5(a), we use a spherical masking field to split the iso-surface into exterior and interior layers. In Figure 6-5(b), we use a planar masking field to split iso-surface into left and right layers.

In Figure 6-6(a), we use a spherical masking field to split the iso-surface of the skull into exterior and interior layers. In Figure 6-6(b), we use a spherical masking field to split the iso-surface of the skin into exterior and internal layers. Then we use spherical texture models to texture the volume skull object and the volume CTHead object.

The top part of the skull and the front part of the face are removed (the opacity is set to zero) to expose the texture differences between different layers.

Tietjen et al. also present volume rendering techniques which combine silhouettes, surfaces, and volume rendering for surgery education and planning [136]. They use z-buffer techniques in their volume rendering engine and the original volume has been segmented in advance.

The depth information and the segmentation are used to enhance the distinction between different objects. In contrast, in our novel pipeline, the depth information and the segmentation can be calculated simultaneously.

6.5 Universal Template Atlas

Given a set of warped texture images (intermediate templates), we can now realistically render a volume object using a variety of semantic constraints during volume rendering.

We propose in this section to use a set of intermediate template texture images, *a universal template atlas*, to store the colour and other necessary information such as illumination, bump mapping and hyptertexture information. The rendered intermediate templates are actually the rendered images from semantically segmented parts of the volume object.

The texture atlas, introduced by Maillot et al. [137], is a set of mesh-based maps. These maps are further packed into a 2D square. The texture atlas can be generated using the following steps: first, partition the 3D mesh model to be textured into a set of parts; second, each part is provided with a parameterisation; third, each unfolded part

(referred to as chart) is packed in a 2D texture.

Mesh partitioning, mesh parameterisation, unfolded chart validation and the packing algorithms which are used to gather the unfolded charts in texture space, are crucial steps in the process of generating a texture atlas. Unfortunately, if a mesh model has sharp edges, then the segmentation methods would generate too many charts, texture manipulation operations become impractical for animators. In addition, cracks and holes on mesh models will lead to texture artifacts on the rendered objects.

Comparing Maillot et al.'s texture mapping methods, our techniques offers the following advantages and novelties:

(1) Given a volume object, there is no need to construct a mesh model as an intermediate model. We render a set of intermediate templates using a variety of semantic constraints, as described in the previous chapters.

(2) Each intermediate template can be a texture atlas itself, which integrates the positions and the geometrical features of different segmented volume parts.

(3) A set of intermediate templates can be a texture atlas of spatially segmented volume parts, in particular self-occluded iso-surfaces.

(4) Each intermediate template can be independently rendered using an appropriate plenoptic or planar model.

(5) The rendering template atlas becomes part of the volume rendering process; therefore, CVG operations can be used during texture composition.

(6) There are no holes and cracks in our universal template atlas. So waste of memory due to complex borders or holes in mesh surfaces of traditional CAD models is not a problem in our system. In addition, we can use the MDS-based LWLS method to warp the template atlas, so the texture can be directly overlaid onto texture atlas.

(7) As explained by Levy et al. [138], some surfaces are too complex to be correctly parameterised. In such cases, triangles may overlap in the parametric space. The MDS-based LWLS method solves this problem by using the MDS boundary condition and linear weighted Laplacian smoothing techniques.

6.5.1 Universal template atlas for texture mapping and annotation

Traditionally, when a texture atlas is used in a 3D paint system, it should meet the following requirements [138]:

(1) The chart boundaries should be chosen to minimize texture artifacts.

(2) The sampling of texture space should be as uniform as possible.

(3) The atlas should make optimal use of texture space.

Associated with the above three requirements, the traditional process of generating a texture atlas is divided into the following categories:

Segmentation: The model is partitioned into a set of charts.

Parameterisation: Each chart is unfolded, i.e., put in correspondence with a subset of $\mathbb{R}^2$.

Packing: The charts are gathered in texture space.

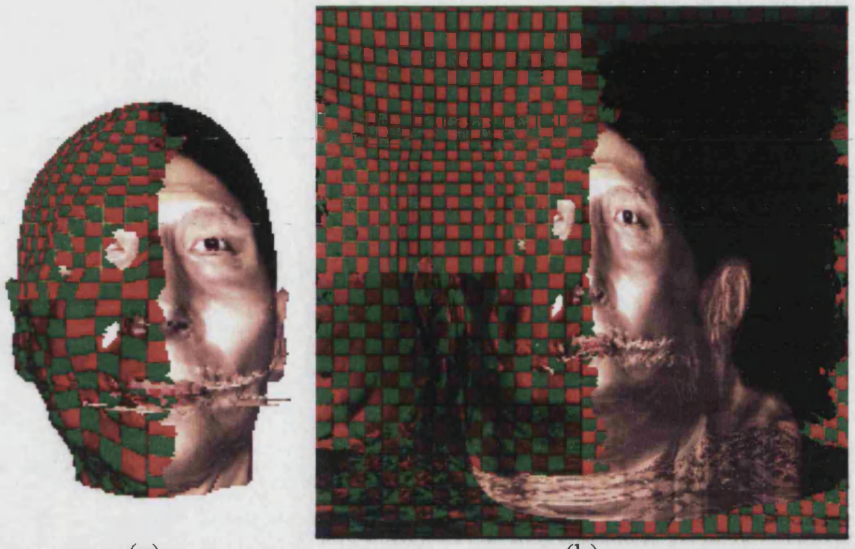
The remainder of this section presents our offered texture atlas method for these three steps, in particular from the view point of volume rendering. We introduce a volumetric based texture atlas generation method, meeting these requirements by using field functions, spatial transfer functions and masking fields. We integrate a variety of volume visualisation techniques. Therefore, our method reduces texture artifacts and provides the flexibility to preserve realism (nature looking phenomena) for volume objects.

In our algorithm, first, the borders of the charts do not need to be fixed. For a single intermediate template, different charts are rendered seamlessly. Self-occluded charts are rendered into separate intermediate templates.

Second, there is no need to worry about the parameterisation overlap [139], where the boundary of the surface self-intersects in texture space.

Third, each chart of the texture atlas can have independent resolution. The intermediate template of a segmented volume part can be generated independently, with an arbitrary resolution [39]. The high-resolution charts are generated through volume rendering, but through interpolation. Therefore, high resolution charts additionally give highly detailed images of geometrical features (shapes, structures, etc.) of volume objects. The examples of the above are given in Figure 6-7.

Segmentation into charts: FF, STF and field masks

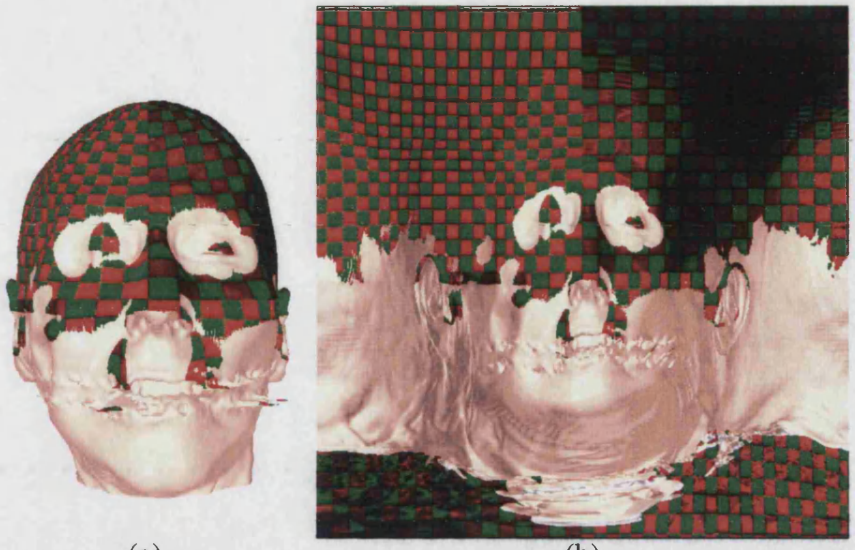


(a) (b)

Figure 6-7: Textured CTHead using projective masking fields: (a) 3D space can be split using planar projection (from left to right). The left layer is textured using a chessboard image. The highlighted area is textured using MDS-LWLS control, whereas the shaded area uses standard spherical indexing control. (b) A universal template atlas can be rendered using a standard spherical template, consolidating the split (left and right) of iso-surfaces with different texture fragments: chessboard and facial image; and spatial constraints: highlighted (left + top constraints) and shadowed (left + bottom constraints) areas.

As shown in Figure 6-7, a universal template atlas can be rendering using a variety of semantic constraints including masking fields, MDS-LWLS constraints, spatial constraints and plenoptic texture mapping models. Different texture charts can be rendered seamlessly into the intermediate template. In addition, texture charts can also be warped using MDS-LWLS constraints, which will guarantee the reduction of shear effects and guarantee high quality multi-resolution texture mappings, in particular, preserving geometrical features.

Chart parameterisation: projective texture models



(a) (b)

Figure 6-8: Textured CTHead using projective masking fields: (a) 3D space can be split using planar projection (from top to bottom). The top layer is textured using a chessboard image. The highlighted area is textured using MDS-LWLS control, whereas, the shaded area uses standard spherical indexing control. (b) A universal template atlas can be rendered using a standard spherical template, consolidating the split (top and bottom) of iso-surfaces with different texture fragments: chessboard and originally rendered image using DVR; and spatial constraints: highlighted (left + top constraints) and shadowed areas.

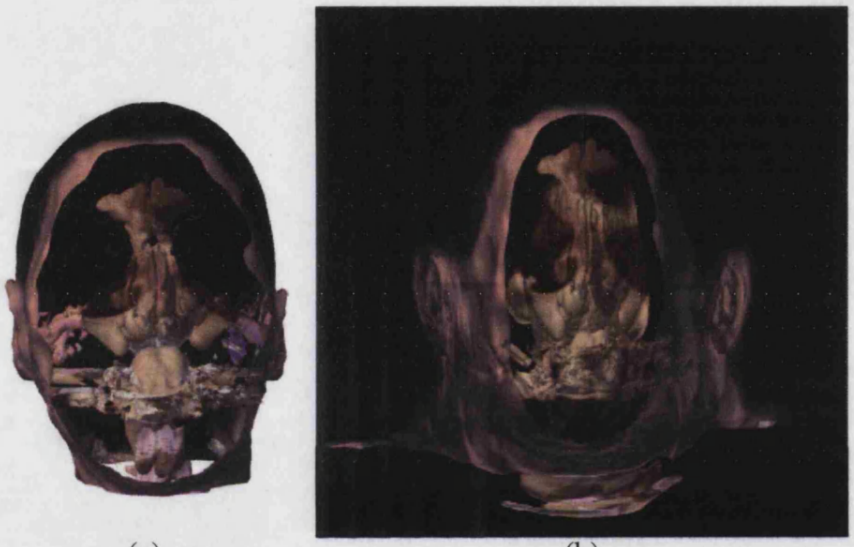
As shown in Figures 6-7 and 6-8, a universal template atlas can be be a useful tool to consolidate different volume rendering information into a universal template. As we will demonstrate in the next example, this special feature of the universal template atlas provides not only the flexibility to texture map volume objects but also to construct the bridges between 2D image models, 2.5D or 3D mesh models, and 3D volume datasets.

Chart packing in texture space: semantic constraints in volume rendering

It is clear that we can put charts from several segmented objects into one intermediate template. The different charts are actually different intermediate templates of individual volumetric objects. As shown in Figure 6-9, figure(a) is the volumetric vi-

sualisation of the skin and the tongue and the ear channels of the CTHead; figure(b) is the rendered intermediate templates of these segmented volumetric objects.

The beauty of this template atlas is, as we will demonstrate in the next section, if we would like to further register 3D features (for instance, tumour transitions during radiotherapy treatment), then we could trace the the 2D projections of these 3D features on the information enhanced intermediate template.



(a) (b)

Figure 6-9: Chart packing into one intermediate template: (a) Segmented iso-surfaces of skin, tongue, and ear-channels. The iso-surfaces (the same level sets) are segmented using masking fields and are textured independently. (b) Different parts of intermediate templates of these segmented objects (charts) are consolidated into one intermediate template, which further provides a landmark guide for 3D feature registration.

6.5.2 Universal template atlas for feature registration: tracking 3D features in 2D space

We use a variety constraints such as field functions, spatial functions, texture mapping functions and colour transfer functions, in our volume rendering engine. Therefore, intermediate templates include a variety of information such as projections of 3D geometrical features, colour information and texture information. As we demonstrate in this thesis, intermediate templates are informatively enhanced by this additional infor-

mation.

Being an information enhanced image, a 2D intermediate template can be a powerful tool for estimating three dimensional deformations (point cloud representation), modelling deformable models, and registering 3D and 2D features (converting 3D feature registration into 2D feature registration). Using MRI or CT slices (greyscale images) for medical registration applications, intermediate templates do facilitate registration processes such as segmentation, positioning, registration, and statistical transition modelling.

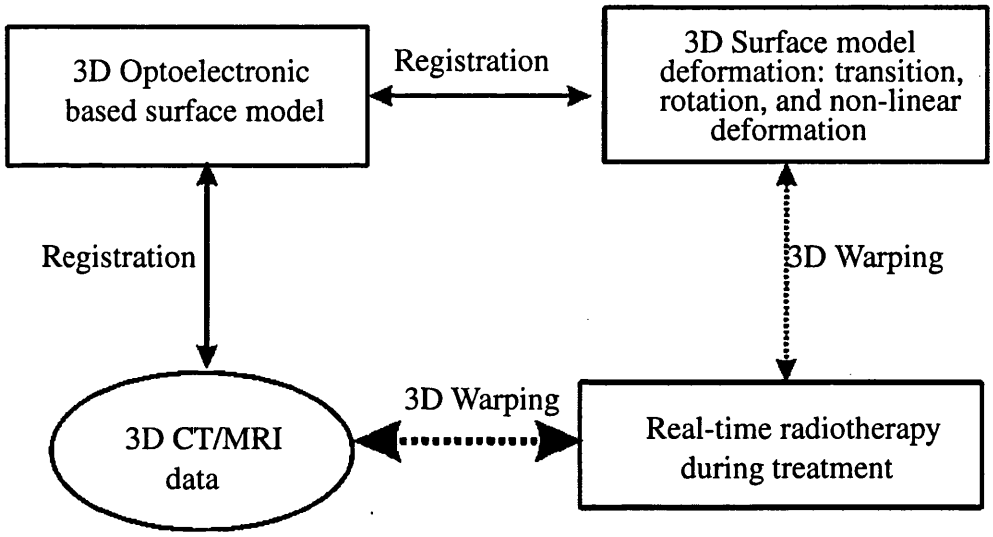


Figure 6-10: Conventional registration process for patient set-up in radiotherapy treatment.

Figure 6-10 shows the conventional registration process in a radiotherapy treatment system (Please refer to the website of the project MEGURAPH, Metrology Guided Radiotherapy, for the technical details [140]).

In order to locate the 3D positions of internal features, for instance, soft tissues or cancers, the exterior surface (skin) of patients' MRI / CT data must be registered in real-time with the 3D reconstructed deformable exterior surfaces of patients, who are under radiotherapy treatment.

After the pre-processing of the registration, the internal organs of the MRI / CT data can be warped according to the current positioning set-up. The position of the warped internal organ or cancer provides guide information for positioning the radio beam.

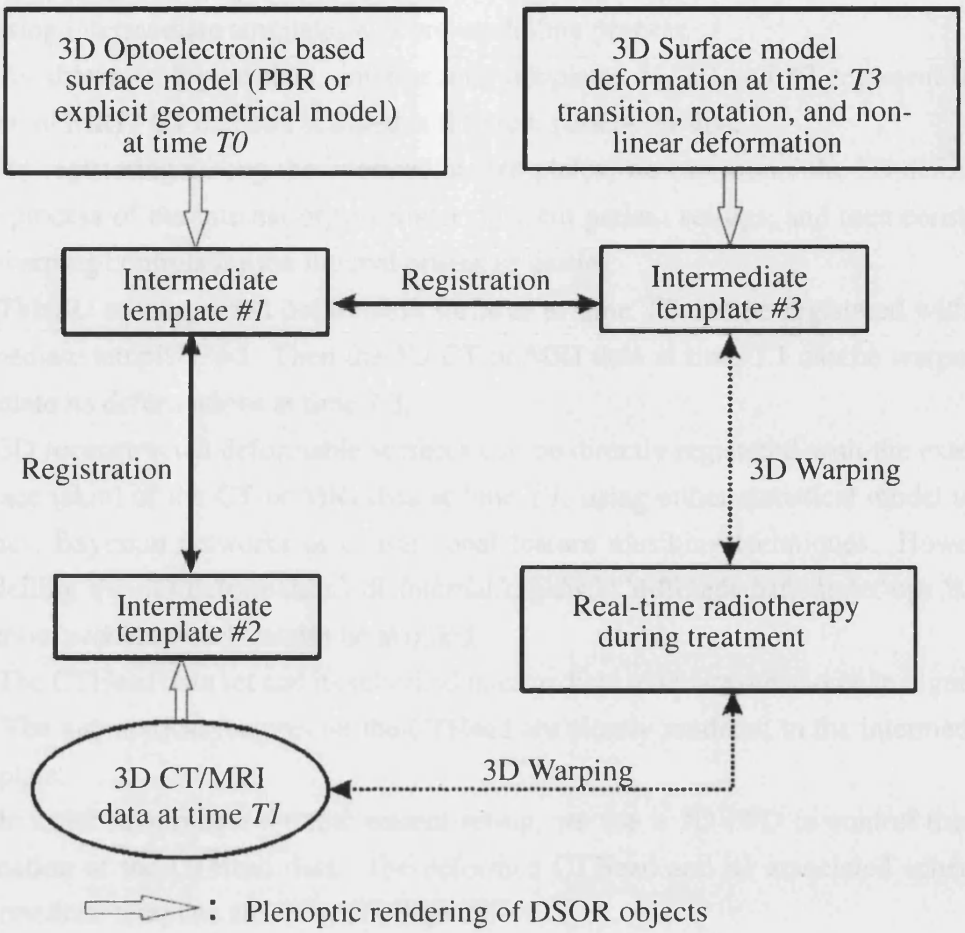


Figure 6-11: Intermediate template based registration process for medical applications.

The above 3D deformable surfaces are traditionally reconstructed using structure light. Such 3D deformable surfaces lose almost all original texture information on the skin. In addition, the reconstructed 3D surfaces are further interpolated or smoothed using 3D spatial filters.

So, if we construct a 3D deformable surface using computer vision techniques, such as shape-from-shading, or by projecting structures light, then the useful geometrical features on the 3D deformable surface will be lost. For instance, texture information such as colour, and wrinkles on the skins will be lost. Therefore detecting and matching feature points becomes very difficult, in particular in those applications with the need for high accuracy registration.

We have already introduced information enhanced intermediate templates for feature registration and matching. The conventional registration process can be improved

by using intermediate templates as a pre-modelling process.

As shown in Figure 6-11, intermediate templates #1, #2 and #3 represent three different MRI / CT datasets scanned at different patient set-ups.

By registering among the intermediate templates, we can model the 3D deformation process of the internal organs under different patient set-ups, and then construct 3D warping controls for the internal organs or tissues.

The 3D reconstructed deformable surfaces at time $T3$ can be registered with intermediate template #3. Then the 3D CT or MRI data at time $T1$ can be warped to simulate its deformations at time $T3$.

3D reconstructed deformable surfaces can be directly registered with the exterior surface (skin) of the CT or MRI data at time $T1$, using either statistical model techniques, Bayesian networks or conventional feature matching techniques. However, modelling the 3D deformations of internal organs at different patient set-ups is the essential process which cannot be avoided.

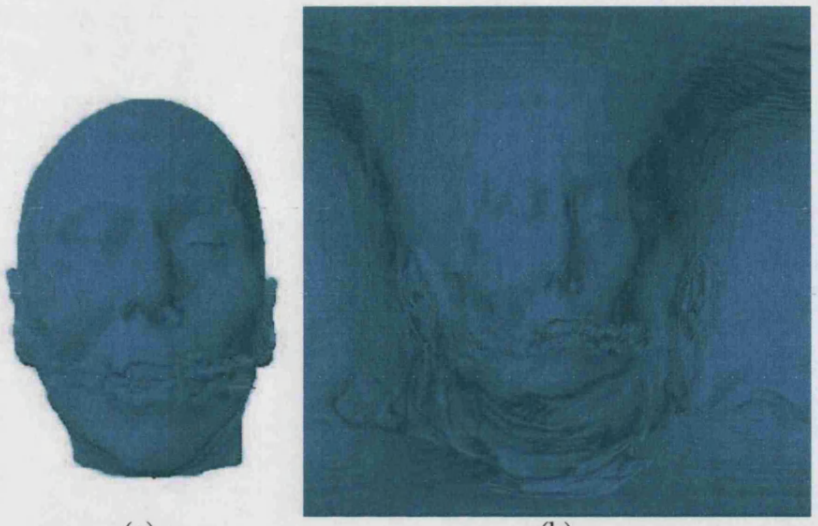
The CTHead data set and its spherical intermediate template are shown in Figure 6-12. The geometrical features on the CTHead are clearly rendered in the intermediate template.

In order to simulate the real patient set-up, we use a 3D FFD to control the deformation of the CTHead data. The deformed CTHead and its associated spherical intermediate template are shown in Figure 6-13.

The soft tissues within the CTHead shown in Figure 6-12 are given in Figure 6-14. The deformed soft tissues within the CTHead shown in Figure 6-13 are given in Figure 6-15.

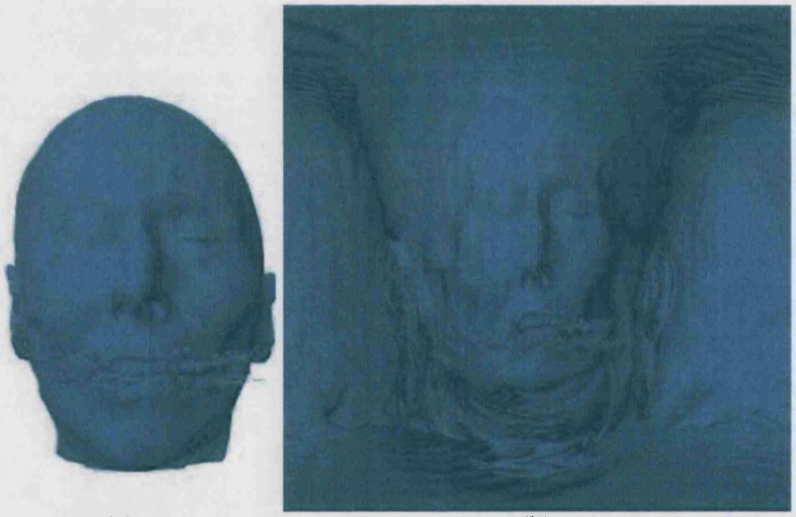
Note that the extent of the deformation of the soft tissues within the CTHead are different to the extent of deformation of the skin.

As we will demonstrate in the following subsections, geometrical features of soft tissues and skins can be detected and tracked on different intermediate templates. These different intermediate templates can be rendered using the same volume data but captured at different patient's set-ups.



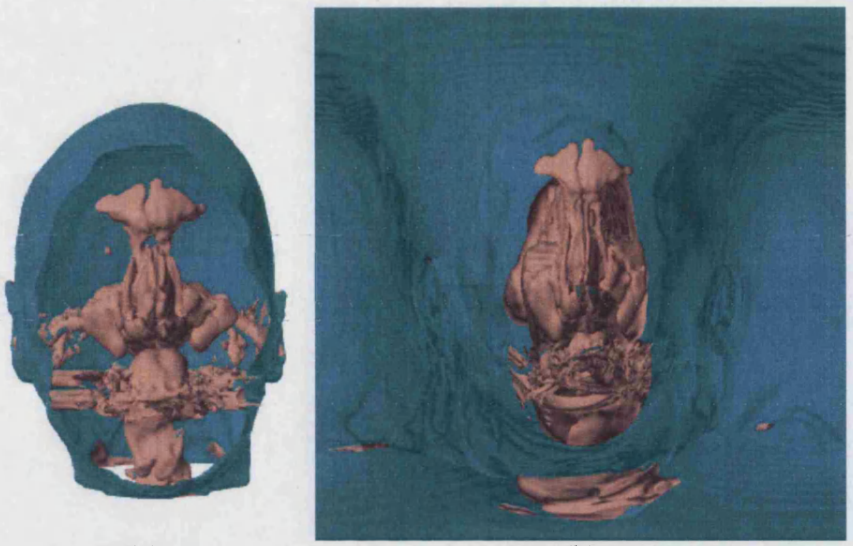
(a) (b)

Figure 6-12: (a) CTHead data set. (b) Its spherical intermediate template.



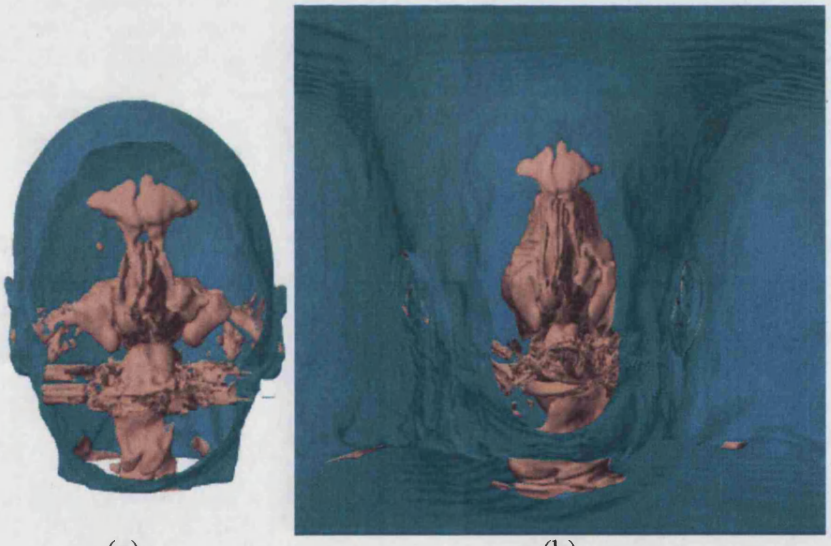
(a) (b)

Figure 6-13: (a) Deformed CTHead data set. (b) Its spherical intermediate template.



(a) (b)

Figure 6-14: (a) Soft tissues within the CTHead data set. (b) Its spherical intermediate template.



(a) (b)

Figure 6-15: (a) Soft tissues within the deformed CTHead data set. (b) Its spherical intermediate template. In (b), the positions of these deformed tissues can be used to register the movements of these tissues during patients' set-ups.

3D FFD deformation lattice

To achieve this deformation, we put a volume object into a flexible parallelepiped lattice structure, which has $4 \times 4 \times 4$ control points. When the parallelepiped is deformed, then the volume object inside will deform with it accordingly. The details of FFD control implemented in VLIB can be seen in Winter's PhD thesis [6].

Given the three orthogonal axes of the parallelepiped lattice, $\bar{S}$, $\bar{T}$ and $\bar{U}$, the origin of the three axes is at p_0, a position with global coordinates in 3D space. Let (s, t, u) be the local coordinates within the parallelepiped lattice, then the $4\times4\times4$ control points, C_{ijk}, can be the control points for a 3D deformation control.

By first calculating the local coordinates (s, t, u) of point p, then the deformed position $\acute{p}$ can be evaluated using the following equation of the tri-variate parametric volume:

$$\acute{p} = \sum_{i=1}^{4} \binom{4}{i} (1-s)^{4-i} s^i \{ \sum_{j=0}^{4} \binom{4}{j} (1-t)^{4-j} t^j [\sum_{k=1}^{4} \binom{4}{k} (1-u)^{4-k} u^k C_{ijk}] \} \qquad (6.1)$$

		$i=1$	$i=2$	$i=3$	$i=4$
$k=1$	$j=1$	(p_1, p_1, p_1)	(p_2, p_1, p_1)	(p_3, p_1, p_1)	(p_4, p_1, p_1)
	$j=2$	(p_1, p_2, p_1)	(p_2, p_2, p_1)	(p_3, p_2, p_1)	(p_4, p_2, p_1)
	$j=3$	(p_1, p_3, p_1)	(p_2, p_3, p_1)	(p_3, p_3, p_1)	(p_4, p_3, p_1)
	$j=4$	(p_1, p_4, p_1)	(p_2, p_4, p_1)	(p_3, p_4, p_1)	(p_4, p_4, p_1)
$k=2$	$j=1$	$(p_1, p_1, p_2+\alpha)$	$(p_2, p_1, p_2-\alpha)$	$(p_3, p_1, p_2+\alpha)$	$(p_4, p_1, p_2-\alpha)$
	$j=2$	$(p_1+\alpha, p_2, p_2+\alpha)$	$(p_2+\alpha, p_2, p_2-\alpha)$	$(p_3+\alpha, p_2, p_2+\alpha)$	$(p_4+\alpha, p_2, p_2-\alpha)$
	$j=3$	$(p_1+\alpha, p_3, p_2+\alpha)$	$(p_2+\alpha, p_3, p_2-\alpha)$	$(p_3+\alpha, p_3, p_2+\alpha)$	$(p_4+\alpha, p_3, p_2-\alpha)$
	$j=4$	$(p_1, p_4, p_2+\alpha)$	$(p_2, p_4, p_2-\alpha)$	$(p_3, p_4, p_2+\alpha)$	$(p_4, p_4, p_2-\alpha)$
$k=3$	$j=1$	$(p_1, p_1, p_2+\alpha)$	$(p_2, p_1, p_3-\alpha)$	$(p_3, p_1, p_3+\alpha)$	$(p_4, p_1, p_3-\alpha)$
	$j=2$	$(p_1+\alpha, p_2, p_3+\alpha)$	$(p_2+\alpha, p_2, p_3-\alpha)$	$(p_3+\alpha, p_2, p_3+\alpha)$	$(p_4+\alpha, p_2, p_3-\alpha)$
	$j=3$	$(p_1+\alpha, p_3, p_3+\alpha)$	$(p_2+\alpha, p_3, p_3-\alpha)$	$(p_3+\alpha, p_3, p_3+\alpha)$	$(p_4+\alpha, p_3, p_3-\alpha)$
	$j=4$	$(p_1, p_4, p_3+\alpha)$	$(p_2, p_4, p_3-\alpha)$	$(p_3, p_4, p_3+\alpha)$	$(p_4, p_4, p_3-\alpha)$
$k=4$	$j=1$	$(p_1, p_1, p_4+\alpha)$	$(p_2, p_1, p_4-\alpha)$	$(p_3, p_1, p_4+\alpha)$	$(p_4, p_1, p_4-\alpha)$
	$j=2$	$(p_1, p_2, p_4+\alpha)$	$(p_2, p_2, p_4-\alpha)$	$(p_3, p_2, p_4+\alpha)$	$(p_4, p_2, p_4-\alpha)$
	$j=3$	$(p_1, p_3, p_4+\alpha)$	$(p_2, p_3, p_4-\alpha)$	$(p_3, p_3, p_4+\alpha)$	$(p_4, p_3, p_4-\alpha)$
	$j=4$	$(p_1, p_4, p_4+\alpha)$	$(p_2, p_4, p_4-\alpha)$	$(p_3, p_4, p_4+\alpha)$	$(p_4, p_4, p_4-\alpha)$

Table 6.1: $4 \times 4 \times 4$ FFD control points: $p_1 = 0.0$,$p_2 = 0.334$,$p_3 = 0.667$,$p_4 = 1.0$, FFD deformation factor $\alpha = 0.1$.

In other words, given a volume space, we can discretise the space by imposing the FFD parallelepiped lattice. Then for each vertex in the 3D volume space, given its

local coordinates (s, t, u), we can transform its position into Euclidean space using the following Bezier volume equation:

$$P_v(s,t,u) = \sum_{k=1}^{4}\sum_{j=0}^{4}\sum_{i=1}^{4} C_{ijk}B_i(s)B_j(t)B_k(u) \tag{6.2}$$

The $4 \times 4 \times 4$ control points are given in Table 6.1. By changing the FFD deformation factor α, we can deform the CTHead under different deformation controls.

Feature detection and tracking

When simulating the deformation of the skin of patients during real-time radiotherapy, rotation, scaling and illumination variations can become common deformation phenomena [141].

However, the features in time-series intermediate templates can be detected according to the strength of the discontinuity properties of the neighbourhood [142]. Therefore we use Harris feature detectors [143] to detect edges and corners in intermediate templates.

Given an image $I(x, y)$ and a window W which centres on (x, y), then the auto-correlation of the image at the position (x, y) can be defined as:

$$c(x,y) = \sum_{(x_i,y_i)\in W} [I(x_i,y_i) - I(x_i+\triangle x, y_i+\triangle y)]^2 \tag{6.3}$$

where $(\triangle x, \triangle y)$ is a position shift of the image $I(x_i + \triangle x, y_i + \triangle y)$. If we expand the shifted image using a Taylor expansion truncated to the first order terms, we get:

$$I(x_i+\triangle x, y_i+\triangle y) \approx I(x_i,y_i) + [I_x(x_i,y_i)\ I_y(x_i,y_i)][\triangle x\ \triangle y]' \tag{6.4}$$

where $I_x(.,.)$ and $I_y(.,.)$ denote the partial derivatives in the x and y directions.

Substituting the Taylor expansion into the auto-correlation equation, we get:

$$c(x,y) = \sum_{(x_i,y_i)\in W} (I(x_i,y_i) - I(x_i,y_i) - [I_x(x_i,y_i)\ I_y(x_i,y_i)][\triangle x\ \triangle y]')^2 \quad (6.5)$$

$$= \sum_{(x_i,y_i)\in W} (-[I_x(x_i,y_i)\ I_y(x_i,y_i)][\triangle x\ \triangle y]']^2 \quad (6.6)$$

$$= \sum_{(x_i,y_i)\in W} ([I_x(x_i,y_i)\ I_y(x_i,y_i)][\triangle x\ \triangle y]']^2 \quad (6.7)$$

$$= [\triangle x\ \triangle y)][\begin{matrix}\sum_W (I_x(x_i,y_i))^2 & \sum_W I_x(x_i,y_i)I_y(x_i,y_i) \\ \sum_W I_x(x_i,y_i)I_y(x_i,y_i) & \sum_W (I_y(x_i,y_i))^2\end{matrix}][\triangle x\ \triangle y]' \quad (6.8)$$

$$= [\triangle x\ \triangle y]C(x,y)[\triangle x\ \triangle y]' \quad (6.9)$$

where matrix $C(x,y)$ reflects the geometrical structure of the local neighbourhood. If we calculate the two eigenvalues λ_1 and λ_2 of matrix $C(x,y)$, then they form a rotationally invariant description. The geometrical facts and explanations are:

(1) If both eigenvalues are small, then the local auto-correlation function is flat; that is, the intensity discontinuity in any direction is very small. The image has a flat area.

(2) If one eigenvalue is high and the other is low, then this is likely to be an edge.

(3) If both eigenvalues are high, then the local auto-correlation function is sharply peaked, which indicates the image is a corner.

(4) The two eigenvalues are proportional to the principal curvatures of the matrix $C(x,y)$. When the trace of the matrix is large there is an edge, and when the determinant is large there is an edge. So the corner strength signal can be [142]:

$$\Phi(x,y) = |C(x,y)| - \kappa Trace^2(C(x,y)) \quad (6.10)$$

where $\kappa = 0.004$ is set empirically.

As previously mentioned, first, we detect good corners using the Harris corner detectors on two intermediate templates. These two intermediates are rendered under different deformation controls which simulate the different patient set-ups. The two detected corner sets are therefore independent of each other.

The detected feature sets in different intermediate templates are dynamically changed. First, the ith feature in an intermediate template at time $t0$ might become the jth feature in another intermediate template at different time $t1$. Second, the ith feature might disappear since a feature existing only at time $t0$ can become less prominent at time

$t1$, due to skin deformation.

The variety of deformation mechanisms such as scaling, rotating, shearing, and the different combinations of these deformation mechanisms, require an extremely robust feature matching algorithm. Such an algorithm will consider not only the correlation factor, but also the human perceptual feature matching mechanisms; that is, grouping, similarity, and exclusivity.

A simple matching algorithm was proposed by Scott and Longuet-Higgins [147]. Their algorithm incorporates both the principle of proximity and the principle of exclusion. Given two feature sets, $i \in M$ and $j \in N$, the distance metric between features can be defined as:

$$G_{ij} = e^{-r_{ij}^2/2\sigma^2} \tag{6.11}$$

Matrix G captures relationships for all possible feature pairs. r_{ij} is the Euclidean distance between features i and j.

The Gaussian weighted distance measure, $G_{i,j}$, is unique. It not only relates the distance measure with feature coordinates but it also scales distance weighting (the discrepancy in human perceptual vision) using the parameter σ, which controls the extent of the interaction between two feature sets. In addition, the distance measure decreases monotonically with Euclidean distance. Finally, for identical images, the distance measures become positive definite.

Scott and Longuet-Higgins' method can be improved by adding the correlation-weighted factor, C_{ij}, the normalised correlation between two image patches centred at the positions of two matched features [148]. C_{ij} varies from -1 to 1, where -1 represents completely uncorrelated patches, 1 represents identical correlated patches. The correlation-weighted proximity of matrix G can be:

$$G_{ij} = (C_{ij} + 1) \times 0.5 \times e^{-r_{ij}^2/2\sigma^2} \tag{6.12}$$

Given the distance matrix G, we can calculate its SVD decomposition, $[TDU] = SVD(G)$. Then we set the diagonal elements in matrix D to 1. The matrix G now becomes the correspondence matrix P, that is:

$$P = T\ I\ U = TU \tag{6.13}$$

The above operation eliminates the singular matrix and thus re-scales data in fea-

ture space. That is, the largest feature in both row and column in matrix P indicates mutual best match (correspondence).

So we would like to detect the best matching features in both intermediate templates and select the best matching set with the minimal disparity. By first using Scott and Longuet-Higgins method to robustly detect good features, we can then match different feature sets and select the matching with the minimum disparity distance as the optimal one. SVD based correspondence matching algorithms cannot achieve both globally optimal solutions and locally optimal solutions. That is, the parameter σ in Equation 6.12 controls the degree of interaction between the two sets of features. A small value of σ enforces local interactions, a large value enforces global interactions. Therefore, we use RANSAC [144] algorithm in our correspondence matching pipeline. The matching correspondence with minimal disparity will be picked up as the optimal solution. Our feature detecting and matching pipeline can be described as:

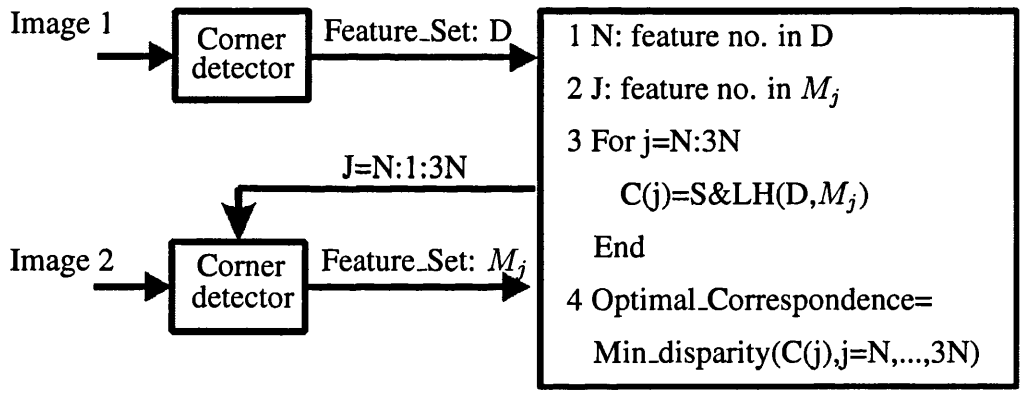
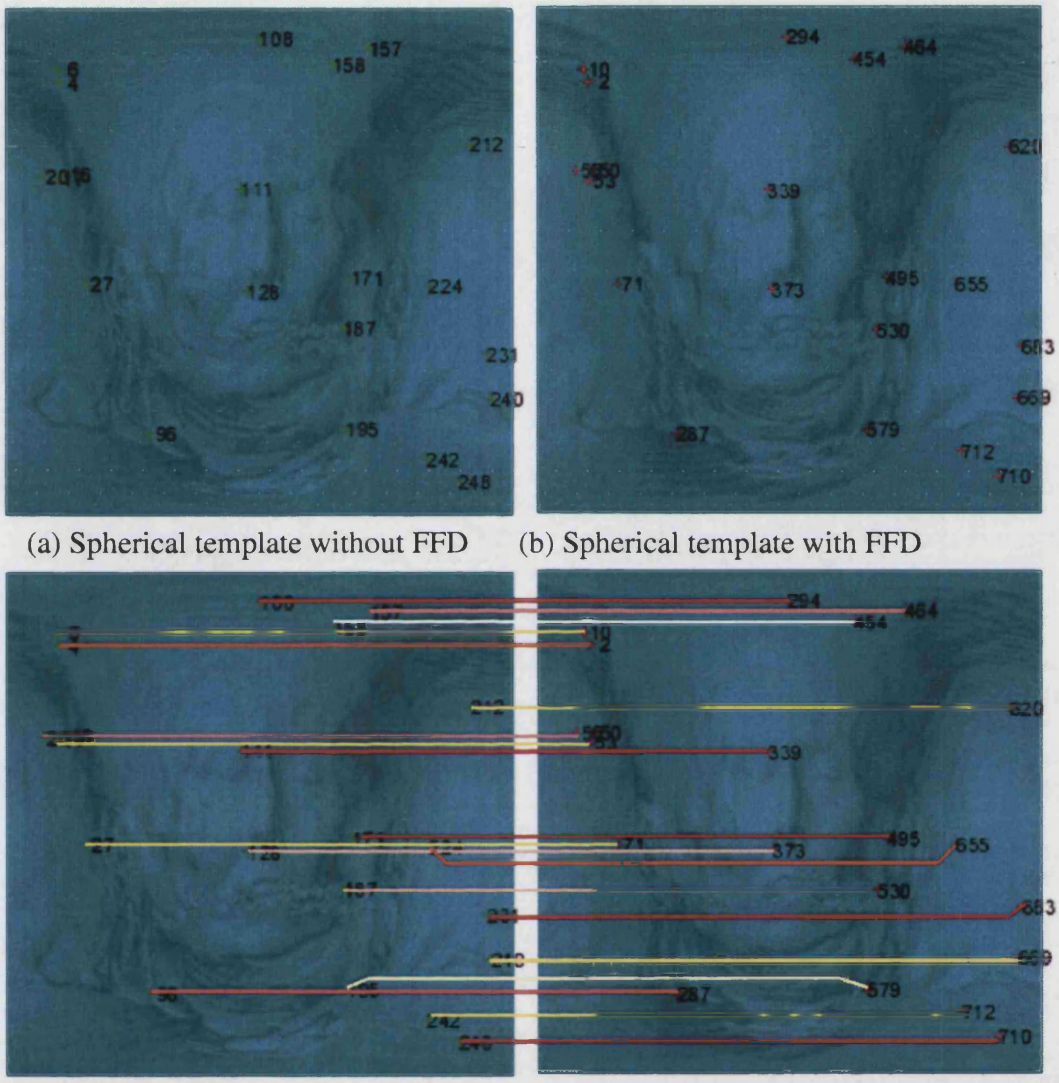


Figure 6-16: Feature detecting and matching in intermediate templates: feature set D and M_j are detected using Harris corner detector. Therefore, the strongest J features in image 2 will be fed into the correlation weighted S&LH matching algorithm. Image 2 is divided into sub-areas. Feature sets are detected in these sub-areas first. The detected feature points are then combined into the feature set M_j. The dynamic feature sets M_j are different to each other in the loop in step 3. The concept of RANSAC is used here. The optimal correspondence solution is calculated by picking up the feature set with minimal disparity between the matching features in M_j.

Note that only the matching correspondences whose correlation coefficients are greater than 0.99 (an empirical threshold) and whose mutual correlation are maximum

at both columns and rows will be picked out as the optimal correspondences. The number of matched correspondences and the accuracy of the matching will directly affect the effect of image warping.



(a) Spherical template without FFD (b) Spherical template with FFD

(c) Matched correspondences in (a) (d) Matched correspondences in (b)

Figure 6-17: Matching correspondences between intermediate template (a) without FFD deformation and intermediate template (b) with FFD deformation. The FFD deformation factor is $\sigma = 0.2$. The index numbers of matching features are different. Correspondences are dynamically detected. Figures (c) and (d) annotate the matching features.

Some experiments

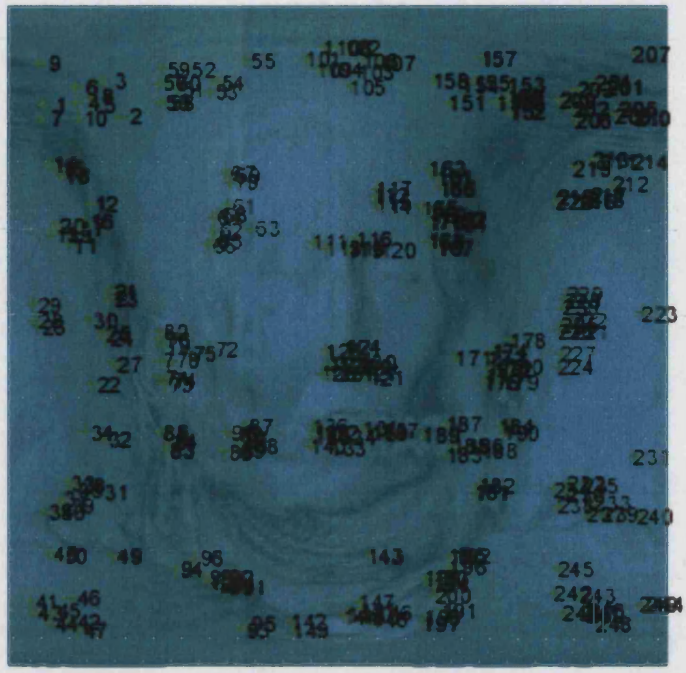
Some results of 2D features detecting and matching are given here. The correspondences in Figures 6-17(a) and (b) are dynamically detected using the Harris corner detector.

The two images are further divided into 5x5 sub-areas. The strongest features in the sub-areas are detected. This mechanism gives us a nearly even distribution of features on the whole images. The total number of features in Figure 6-17(a) is $10\times5\times5$. The total number of features in Figure 6-17(a) is $29\times5\times5$. The distribution and location of these two correspondence sets are given in Figure 6-18. In other words, features are detected locally, whereas, correspondence features are matched globally.

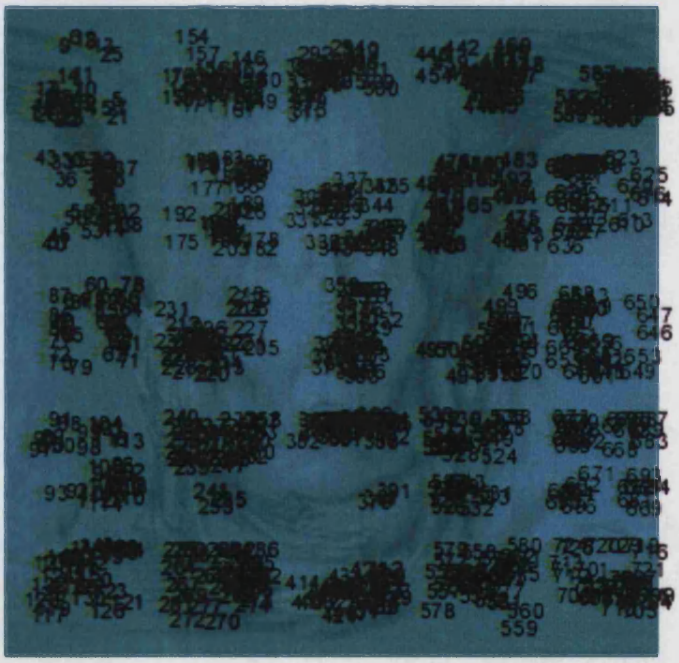
As previously discussed, intermediate templates can be rendered with enhanced information such as enhanced geometrical features, enhanced texture images. They can also be rendered using multi field functions and spatial functions. Therefore, intermediate templates can capture a variety of different information which is well-suited to real applications. Here, we have demonstrated that the feature points of 3D deformable models can be detected and matched using intermediate templates. In addition, the geometrical features on the soft tissues within the volume data set can be detected and matched using the same techniques we provide.

As shown in Figure 6-19, the feature points of soft tissues and the feature points on the skin can be detected and matched at the same time. The matched feature correspondence can be the control points used in fine registration algorithms, such as Lu et al.'s coarse to fine registration techniques [145].

The corresponding features are dynamically detected and matched. In other words, they are the best (the strongest) matching features. These features can always exist during the period of the transitions of different patient set-ups. Therefore, they can act as the control points for investigating the relationship between the model of the exterior surface and the model of the internal organs. Note that conventionally, these control points that act as land marking labels (functional markers) are stuck onto the skin of patients [146].

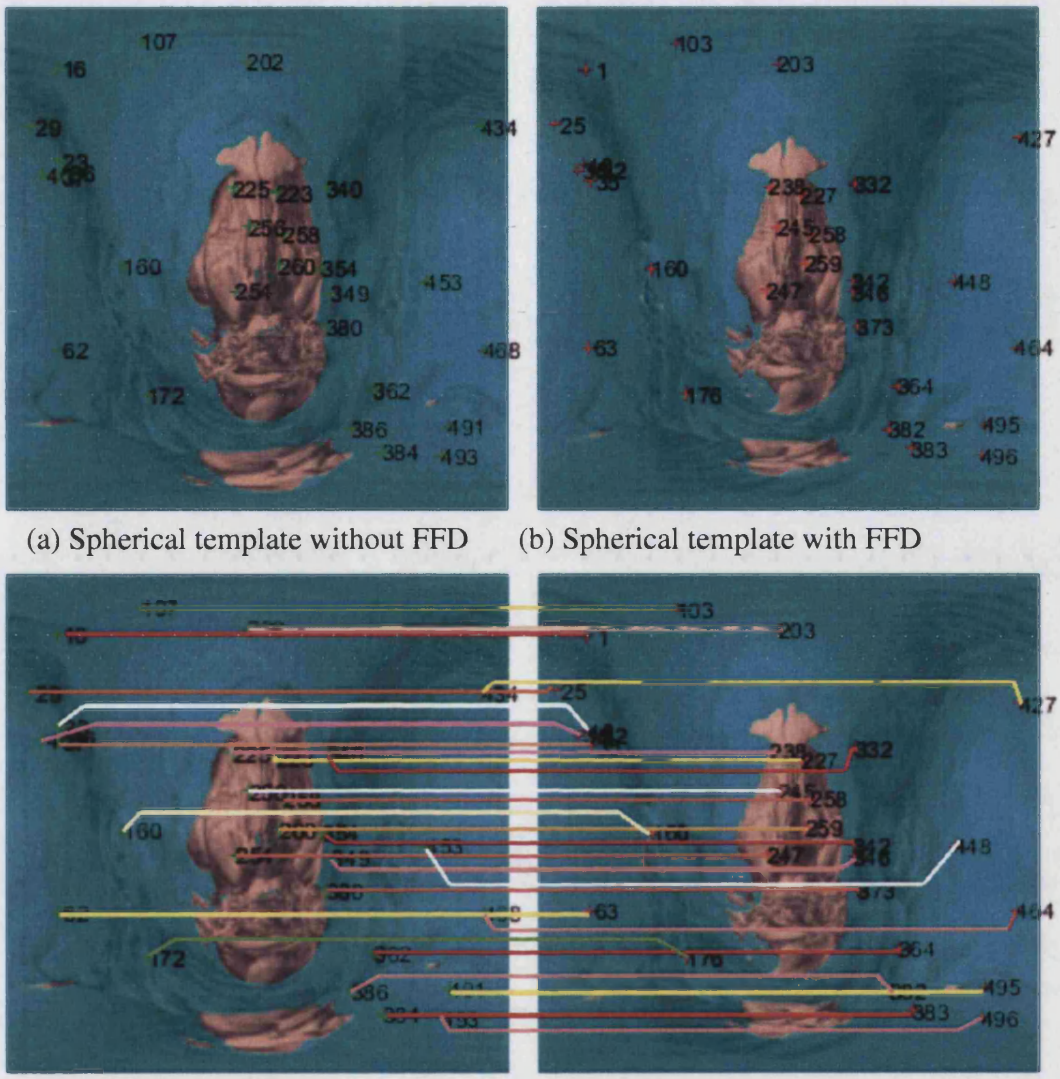


(a) Detected features on the intermediate template without FFD.



(b) Detected features on the intermediate template with FFD.

Figure 6-18: Detected correspondences in intermediate templates (a) without FFD deformation and intermediate template (b) with FFD deformation. The FFD deformation factor is $\sigma = 0.2$.



(a) Spherical template without FFD (b) Spherical template with FFD

(c) Matched correspondences in (a) (d) Matched correspondences in (b)

Figure 6-19: Matching correspondences between intermediate template (a) without FFD deformation and intermediate template (b) with FFD deformation. The FFD deformation factor is $\sigma = 0.2$. The index numbers of matching features are different. Correspondences are dynamically detected. Figures (c) and (d) annotate the matching features.

6.5.3 Computational Expenses

The volume splitting algorithm presented in this chapter is implemented using c/c++. The complexities of the volume splitting algorithm come from the fact that each vol-

ume integral elements of tracing rays needs to calculate its own indexing label by casting an additional tracing ray. Feature detection and tracking algorithms presented in this chapter are implemented using Matlab, since these operations can be easily implemented using matrix operations.

Rendering Figure 6-6(b) take 1 minute 48.3 seconds. The image size is 198x300 pixels and the running step length is set to 0.1. There is one point light in the rendering scene. The size of the CT-head dataset is 180x113x237 voxels.

It is worth pointing out that different configurations of volume rendering pipelines will lead to different complexities of rendering algorithms. The elements of a typical volume rendering pipeline may include: resolutions of rendering images, step lengths of tracing rays, depth of shadow rays, a variety of volume deformation functions, spatial functions, field functions, a variety of interpolation techniques, etc. We use VLIB as our basic volume rendering engine. A detailed discussion about the structural complexity of VLIB was given by Dr. Winter in his PhD thesis [6]. (*The structural network of interconnected data structures in VLIB implementation is given in Appendix C in this thesis.*)

6.6 Conclusion

In this chapter we have introduced a novel constraint for splitting volume objects. We employ the concept of traditional and contemporary z-buffer techniques and we implement a novel scalar field, masking field, to split volumetric space. Our methods can be volume rendering techniques, and additionally with a novel application issue.

We presented a tracing ray based masking ray method. It is based on the volume rendering techniques. Moreover we have extended the algorithm to more general semantic constraints.

Additionally, we have presented an efficient universal template atlas method for texture mapping a volume data set.

The iso-surfaces we use here are all single value level-sets with self-occluding geometrical structures. We can effectively split and annotate different parts of the iso-surface with different texture information.

The user has to specify the indexing number of the masking field to be textured. More research needs to be done on computational cost since we fire extra rays to calculate the masking index number at each position the rendering ray traverses.

Using masking fields and volume visualisation techniques, we can render informatively enhanced intermediate templates. We demonstrated the application to image registration, feature detection and matching. In particular, relational information conveyed by proximity matrices of adjacency correspondences are considered. Therefore, we provide a possible solution for modelling deformable models for positioning patient set-ups in radio-therapy treatment.

The main idea of our intermediate template based registration method is using the distribution of eigenvalues to achieve globally optimal solutions.

Chapter 7

Conclusions and Future Work

This thesis presents an image based texture mapping pipeline for volume objects. The solutions offered touch the problem of texture mapping a volume data set: to find an appropriate representation of digital images that provides a link between 2D, 2.5D, 3D texture models and the 3D volumetric data set.

Multi-layers of iso-surfaces (hidden structures) can be textured independently. Therefore, we believe our methods can be a possible solution to other image-related applications in the areas of volume visualisation. We recognise the importance of giving a volume data set natural appearance and this is the motivation of our projective texture model.

By using MDS-based LWLS flattening control, images can be directly overlaid onto intermediate templates for texture mapping. There is no tangling of textures. Efficient and effective practical applications that need realism based volume objects, such as medical training, surgical planning, can be drawn from this model.

We studied the texture smearing and texture penetration problems in volume visualisation. We presented a survey of discretely sampled object representations. We combined projective texture models with colour transfer functions, field mask constraints, flattened surfaces, and a variety of field function and spatial transfer functions.

7.1 Contribution

We are trying to link textures and 3D volume objects that lack geometrical, topological and semantic constraints. We offer a projective texture model for texture mapping volume objects. This model allows various volume visualisation applications to use this

texture representation and to recover continuous texture colour intensities from discrete image data samples. We have examined several applications such as 2D image based texture mapping, 2.5D pseudo-solid textures, and data-dependent interpolations (DDT) of texture mapping volume objects and other applications of texture mappings in continuous space. The simplicity of the underlying texture model leads to simple and effective applications in these different areas. In particular, we use field masks to control texture penetration, which is often the most obvious challenge of using projective texture models.

Conventional projective texture models index / penetrate textures (colours) along the projective directions. The drawback of these conventional projective based approaches is the shear effect. Textures will not only smear over a relative large area but also penetrate to their neighbours that reside on the projective directions. This thesis provides the field mask solution to this problem. In addition, we demonstrate the flexibility of our semantic texture models by rendering intermediate templates into a universal template atlas with enhanced information. We believe we offer a direction to representing volume objects using such intermediate templates. The flexibility of rendering intermediate templates is given in this thesis.

Our model does not try to detect or attempt to find an explicit geometrical model for the local geometry. It simply probes 3D positions where tracing rays terminate. By doing this, the geometrical structures of the volume data set can be well presented.

We propose an MDS-boundary based LWLS technique to explore the hidden structures of 3D point clouds using 2D flattened configurations. The main strength of this model is that it represents the flattened 3D surface in the plane, without twisting and tangling. The algorithm is based on linear calculations and is thus simple and effective for volume rendering applications.

At heart, out model constructs the connection between pixels and voxels through different techniques such as projective modelling, volume rendering and effective field masking. In particular, we offer a MDS-boundary based LWLS method to eliminate the tangling and twisting of points in 2D configurations.

Projective texture models, MDS-based LWLS flattening algorithms, and field masks are three cornerstones. We have used these three models in the framework of volume visualisation. The details of these methods are given below.

7.1.1 Projective texture models

We present an approach to texture mapping volume datasets. The approach is based on multiple constraints and continuous space mappings to ensure good image quality. The method was composed of three parts: *semantically generating intermediate templates*, *selectively forward and inverse indexing*, and *volume rendering*. This three-part aspect additionally allows the texture image to be independent of the volume data. We demonstrated an extension to 2.5D textures, extruded through the volume, using an approach consistent with 2D texture. A data-dependent triangulation method is used to retain edge quality in texture images. In addition, we presented a colour transfer technique, in which the colour information of pixels in illustration images is transferred to the colour fields of voxels in 3D space.

7.1.2 MDS-based LWLS flattening algorithms: point clouds' configurations and shear effect

We discussed the challenges of our projective texture model such as texture smearing, penetrating and self-occlusion. In order to reduce the shear effect of the projective texture model, we presented a method for smoothing the point cloud within volume objects. The method is based on the *classic multidimensional scaling (MDS)* using shortest-path proximities.

By mapping 3D points into 2D flattened (Euclidean) domain, the presented methods not only flatten the 3D surfaces of the volume object but also preserve information about local geometrical features on the surfaces of volume objects. Our plenoptic based dimensional reduction methods combine three areas: graph layout, volume rendering, and multidimensional scaling. By flattening the 3D points cloud using classic metric MDS, our method benefits the projective texture model by overlaying texture images onto 2D flattened surfaces, i.e., warped intermediate templates, of 3D volume objects.

In order to eliminate the existence of tangling and twisting in MDS configurations, we introduce MDS-based Linear Weighted Laplacian Smoothing model to flatten 3D point cloud within volume objects. This method integrates the benefit of Laplacian smoothing methods and the advantages of classic metric MDS methods. We demonstrated that the proposed method prevents the tangling of flattened points cloud in the application of texturing volume objects.

7.1.3 Field masks and universal template atlas for annotating volume objects

We presented a solution to the texture self-occlusion problem and the texture penetration problem, by splitting the 3D space into differently labelled layers. Using labelled 3D space masks, self-occlusion and texture penetration can be effectively controlled using volume rendering techniques, such as DVR and DSR. We do not build any static field masks in advance. Similar to the techniques of constructing 2.5D pseudo-solid texture models, we dynamically construct field masks during volume rendering. As we demonstrated in this thesis, different iso-surfaces can be split independently using different plenoptic-based field mask models.

So, in our volume rendering engine, the volume data can be textured using different projective texture models, using their associated field masks models, and using a variety of field functions and spatial transfer functions. In particular, a flattened (warped) intermediate template can be used during volume rendering. These operations reduce the effort of texture warping. In other words, texture images can be directly overlaid onto flattened intermediate templates. The texture distortion due to plenoptic projection is reduced.

7.2 Future Work

There are a number of directions in which the work of this thesis can be continued.

2.5D Registration

We offer a possible solution to an medical application: volumetric data registration during the process of radiotherapy treatment. As discussed in Chapter 2 and in [1], volumetric object registration is commonly based on tracking 3D features of segmented objects, either using geometrical information (static or statistical models) or advanced pattern recognition and clustering techniques. However, we demonstrated that 3D feature tracking can be converted from 3D space into 2D space, i.e., by:
(1) Rendering plenoptic intermediate templates using additional / enhanced information such as colours and textures; (2) Warping intermediate templates using MDS-LWLS control, which is equivalent to flattening 3D surfaces of volume objects; (3) Tracing 3D features is therefore converted into tracking 2D features in the warped intermediate templates (images).

Conventional 2D registration techniques for medical images are based on feature detection techniques, which use geometrical information and intensity information. In the rendered intermediate template, we can enhance geometrical features by adding colour or texture information through field transfer functions. Detecting and tracking 2D features in intermediate templates become easier and more accurate.

Realistic Volume Graphics

We noticed that MRI scanned, CT scanned, 3D X-ray scanned, and distance scanned data sources such as animals, human bodies, vegetables, and art works, etc., originate from the variety of the surrounds in our daily life, but are conditionally exposed to us without natural appearances.

Therefore, it would be extremely desirable to develop technology for physically realistic volume graphics. It would be even more desirable if the realistic volume characters can be used in medical training and surgical planning. The ability of realistic volume graphics will innovate the basic research framework of volume graphics.

Estimating optimal proximity in MDS, and optimal linear weights in LWLS

We use Euclidean distance, geodesic distance, and MDS based linear Laplacian weights to smoothing 3D point clouds. The shape of the flattened MDS boundary is affected by the estimation of proximity. Different proximity estimation will affect the performance of MDS smoothing.

Using local neighbourhood information to construct low-dimensional-scaling models is our main consideration. We will focus on using local multidimensional scaling (LMDS) plus graph layout techniques. We will construct desirable drawings of graphs by balancing attractive forces between near points and repulsive forces between distant points. The LMDS method is investigated by Chen and Buja using nonlinear reduction models, in the draft [127] in 2006. In this thesis, we are facing the same problem, in particular for volume based applications. However, we prefer to use linear dimension reduction methods. We are currently working on this.

Visualisation of Errors

We use direct surface rendering algorithm and direct volume rendering algorithm in our system. While we are trying to locate the positions of iso-surfaces using different step-lengths of tracing rays, we find that the iso-surfaces' termination positions are different. This means that we can not exactly locate the positions of iso-surfaces. In other words, visualisation errors do exist and may not become tolerable under some circumstances, in particular in surgical planing [99]. Therefore we would like to in-

vestigate and quantify visualisation errors to improve the visualisation accuracy in our system.

7.3 Conclusion

In this thesis, projective texture models are developed to link 2D images and 3D volumetric DSORs. MDS and LWLS methods are combined to deliver a working and robust method for smoothing point data and to explore hidden 3D structures of volume data. MDS based LWLS techniques are powerful for smoothing 3D surfaces and are effective for other texture mapping applications. In addition, a novel splitting model for volume objects is developed to eliminate texture penetrations.

We are trying to texture mapping volume objects with realistic appearances. This leads to the research which presents an effective model (the intermediate template) to exploit properties of 3D features in the information enhanced images. It is not only a better representation of 3D point clouds but also a more effective model than conventional 3D procedural texture techniques. Therefore we believe it is generic for different kinds of texture based applications, in particular for preserving realism for volume objects.

Field functions, spatial transfer functions, field masks, and so on, can be effectively integrated into our rendering process. Therefore, intermediate templates can be rendered flexibly. The enhanced information in intermediate templates includes texture, colour and geometry features, which will further facilitate the processes of feature detection and tracking.

Parts of this research have been presented at the following conferences:

- The 26th Eurographics Conference (Dublin, Ireland, 2005) [1].
- The Institute of Mathematics and its Applications, Vision, Video and Graphics (Edinburgh, 2005) [39].
- The 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, ACM SIGGRAPH (Dunedin, New Zealand, 2005) [40].

Appendix A

Glossary

3D texture mapping - textures can be added to a 3D object using a three dimensional texture block. It can be treated as 3D volumetric sculpture. The 3D object can be effectively carved out of the 3D texture block.

A

Adaptive (early) ray termination - a technique to accelerate the volume rendering. The casted ray can stop sampling the volume object when the accumulated opacity, or intensity, reaches a predefined threshold. The amount of voxels which need to be processed can be reduced.

Aliasing - a problem about objects edges are represented in the ways of discrete representations. The edge is decided by judging whether it is inside or outside of the object. This leads to a jaggy or zigzag effect at the edges.

Annotating volume objects - a process in volume visualisation and the surgical planning application. 3D space of volume object is split and volume object is segmented in advance. Transfer functions or are used to assign colour information to inner structures or segmented and classified geometrical features.

Antialiasing - the process of filtering so that sharp edges can be softened. The geometrical features of edges can be used to facilitate the filtering operation.

B

Backward projection - a volume rendering technique which cast rays through the image plane, into the volume object.

C

Colour transfer - a set of transfer functions which assign colour information to the numerical values or positions in a volume object.

Compositing - the process of merging together all the colours and opacities. The composition can take place in a back-to-front manner.

Constructive volume geometry (CVG) - a constructive representation of volume objects, including both volumetric datasets and scalar fields. The combinational operations are normally defined in the real domain and enable the construction of complex volume objects through the combination of the geometrical and physical properties of simple (solid or amorphous) volume objects.

D

Direct surface rendering (DSR) - a method used to display a surface (level sets) which is inside a volumetric object without constructing a mesh model. The method is based on ray casting and uses tri-linear interpolation to determine the location of the iso-value of the surface.

Direct volume rendering (DVR) - rendering an image from a volume object without any intermediary step in which a surface is generated. Volumetric ray tracing is referred as direct volume rendering in this thesis.

Distance field - a volume dataset in which the voxel values give the approximate distance to the surface of interest.

E

Environment mapping - a texture mapping method which reduces the computation of a reflection ray to a simple intersection. A scene is mapped onto a cubic or a spherical or cubic map.

F

Forward projection - the name given to a collection of volume rendering algorithms which synthesis an image by projecting the voxels onto image plane.

I

Illustrating volume objects - a process of volume visualisation which combining the familiarity of physical based illumination model or non-photo realistic rendering tech-

niques to make the important visual cues become more prominent. Volume illustration provides a flexible unified framework for enhancing structural perception of volume models through the amplification of features and the addition of illumination effects.
ISO-surface - a surface of equipotential. For a field function $f : \mathbb{R}^4 \longrightarrow \mathbb{R}$, the τ-iso-surface, $S(\tau)$, is defined as $x^d : f(x^d) = \tau$, τ is referred to as an iso-value or threshold.
ISO-value - a value which represents something interesting in a particular domain. In 3D iso-surface, the iso-value defines the surface of interest and is calculated such that all points on the surface (level sets) have a function value equal to this value.

M

Marching cubes algorithm - an algorithm for determining an iso-surface (mesh model) in a 3D dataset. Function values are evaluated at regular discrete points to make a 3D grid of data values (voxels). Eight neighbouring voxels make up a cube, from which the surface can be determined using lookup tables for each of the possible cases. Since each of the eight voxels can be inside or outside of the surface, there exist 256 cube configurations, which ban be reduced to 14 configurations using symmetry. The lookup table indicates the triangles to be added to a triangular mesh, with their vertices interpolated from known voxel positions and values.

N

Non-photorealistic rendering - an approach to image synthesis which gives the ability to use various effects which do not attempt to realistically model and render the scene, while still retaining the advantages of a conventional rendering system.

O

Object self-occlusion - parts of an object hidden from the viewer

P

Parameterised texture mapping - adds surface details by wrapping a two dimensional texture map around an object.
Projection mapping - adds surface detail in two stage. The texture is first mapped onto a single (intermediate) three dimension object (plenoptic model), S mapping. The intermediate object is then mapped onto the object being rendered, O mapping.

R

Ray casting - the process of sending a primary ray from a point in space into a scene with direction. The ray terminated and the intensity is accumulated according to shading models.

Ray tracing - the process of sending an additional second ray of ray casting process. Upon the intersection with an object, the second ray will be spawned in order to determine all contributions of other light sources.

S

Semantic functions - Transformation functions could be extended to a generic description, that is: *semantic functions*, which operate on space criteria, logical criteria, temporal criteria, geometrical criteria, topological criteria, and so on.

Semantic transfer function - A semantic transfer function operates on both spatial position p and field value $F_i(p)$; whereas, a transfer function can only manipulate field values on the same position, spatial transfer function can only transfer the same scalar value from one position to another.

Scalar field - a scalar field takes a coordinate and returns the value (interpolated or actual) at that coordinate.

Solid texture - Complex three dimensional textures constructed from primitive, non-linear and basis functions.

Space partition - the process of dividing space into smaller subspaces. The reasoning behind this is that processes can be carried out on different subspaces independently.

T

Texel - the individual pixels of a two dimensional texture image. Texel is also the name given to the element of a three dimensional texture block which is used in the volumetric texturing mapping.

Texture image (map) - a two dimensional image which represents textures.

Texture mapping - gives the impression of surface detail by mapping texels onto the surface (level sets) of volume objects during volume rendering.

Transfer function) - a mathematical function which assign different scalar properties to the numerical values of the volumetric object.

Two-part texture mapping - see *projection mapping*.

V

Volume graphics - a sub-field of computer graphics which models and renders both discrete volumetric datasets and continuous scalar fields. The main aim of this field is to provide all known graphics effects and techniques within the volume environment. Typical operations include CVG, volume sculping, hypertextures, etc.

Volume rendering - a process for obtaining images from three dimensional volume objects. Volume objects can be represented by a set of scalar fields in a way that rays can pass through the 3D space of these scalar fields simultaneously. Typical scalar fields include colour components, opacity settings, distance field and velocity field. Direct surface rendering and direct volume rendering techniques are two typical algorithms used in this thesis.

Volume sculpting - a free-form, interactive modelling technique based on sculpting voxel-based solid material. It is also used to explore the inner structure of a dataset by removing material step by step.

Volume visualisation - a sub-field of scientific visualisation focused on visualising and exploring the inner structures of volume objects.

Z

z-buffer - an array $z(x, y)$ of depths for each pixel in images.

Appendix B

Generalised 2D Tutte embedding

To prevent tangling and flipping of internal nodes, we use sufficient conditions given by generalised 2D Tutte embedding algorithms [133]:

Consider the graph G_M of a mesh $M \in M_B$, with N vertices. There exist a boundary (cycle) B in the graph G_M and all other vertices within B are internal vertices. There are n internal vertices, where $n < N$.

Assume boundary B is a convex polygon in $\mathbb{R}^2$. For each internal vertix i, its weights w_{ij} can be constructed using the following conditions:

$$w_{ij} > 0, (i,j) \in E(G_M), \text{ and, } w_{ij} = 0, (i,j) \notin E(G_M) \tag{B.1}$$

$$\sum_{j \in N_i} w_{ij} = 1 \tag{B.2}$$

where N_i is the set of vertices neighbouring the internal vertex i.

Given w_{ij}, internal vertex can be calculated using the following equations:

$$u_i = \sum_{j=1}^{N} w_{ij} u_j \tag{B.3}$$

The above equation can be written as the following linear system:

$$u_i - \sum_{j=1}^{n} w_{ij} u_j = \sum_{j=n+1}^{N} w_{ij} u_j \tag{B.4}$$

The above equation can be written into the matrix form:

$$Au = b \tag{B.5}$$

$$A(i,j) = \begin{cases} -w_{ij}, & if\ i \neq j \\ 1 & if\ i = j \\ 0 & iff\ (i,j)\ not\ in\ E \end{cases} \tag{B.6}$$

where u and b are positions of internal vertices and boundary vertices. Matrix A is an M-matrix therefore it is non-singular. So the above linear system has unique solution. Given matrix A and boundary positions b, we can calculate the positions of internal vertices u.

Lemma 1: Every internal vertex i is positioned at u_i, and it is the solution of the above linear system. Then every internal vertex lies strictly inside the convex polygon Q with vertices at $b = \{u_{n+1}, u_{n+2}, ..., u_N\}$.

Theorem 1: The solution of the above linear system yields a planar embedding of the graph G_M of mesh M, in which no triangles overlap.

The details of the approvement of Lemma 1 and Theorem 1 are given in [133] on page 47 and page 48.

Appendix C

Structural Complexity of VLIB

The complexity of volume rendering pipelines highly depends on different configurations of different applications. The following diagram shows the network of interconnected data structure in VLIB implementation given by Dr. Winter in his PhD thesis [6].

Each data structures are represented using a box. They can be a rendering scene, or some elements of a rendering scene. Their data structures are implemented using "struct" structures in C. Arrows from one box to another box construct *dynamic* links in different configurations of different volume rendering pipelines. The functions are implemented using pointers in C.

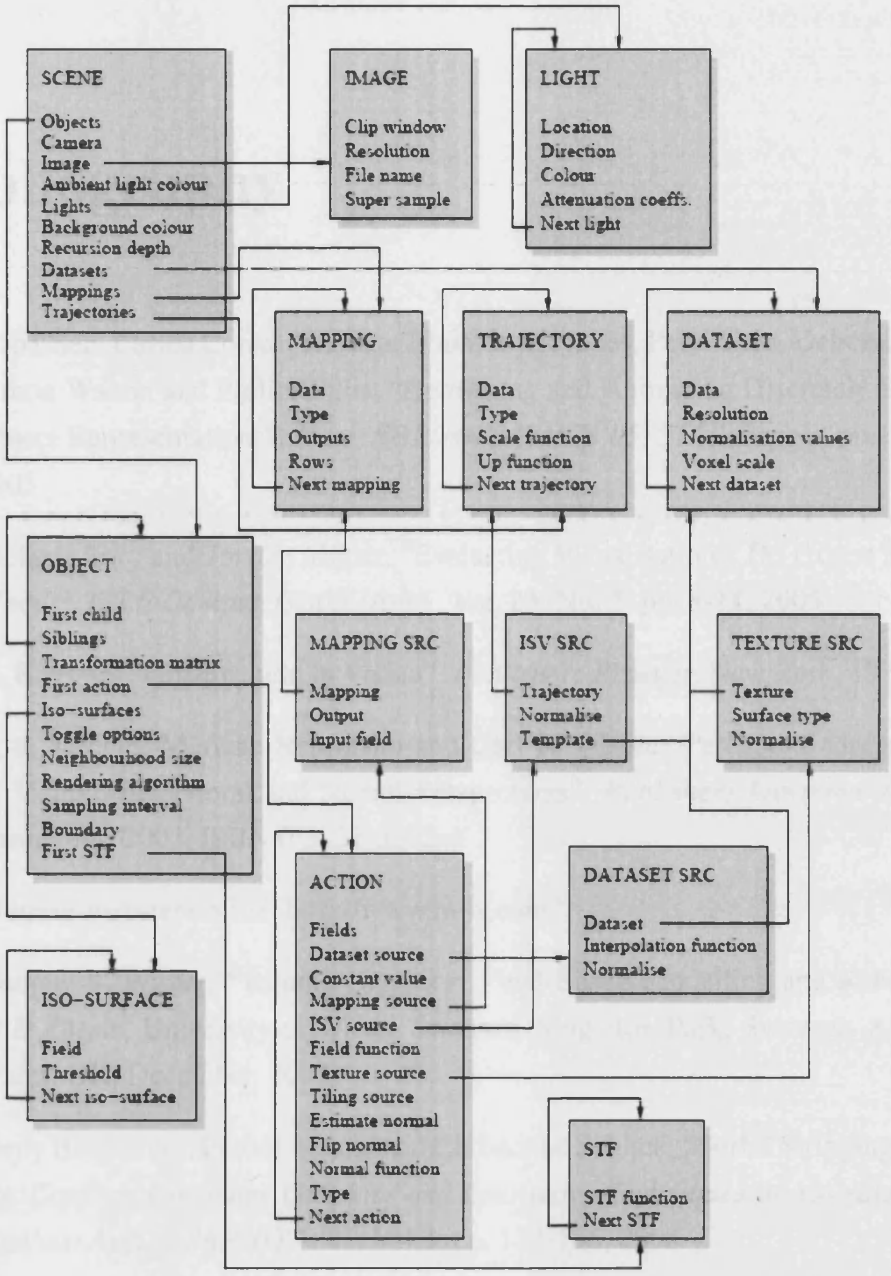


Figure C-1: Network of interconnected data structure in a VLIB implementation [6].

Bibliography

[1] Min Chen, Carlos Correa, Shoukat Islam, Mark Jones, Peiyi Shen, Deborah Silver, Simon Walton and Philip Willis, "Deforming and Animating Discretely Sampled Object Representations", *Proc. EUROGRAPHICS'05, STAR Report*, pp.113-140, 2005

[2] Melanie Tory and Torsten Möller, "Evaluating Visualisations: Do Expert Reviews Work?", *IEEE Comput. Graph. Appl.*, Vol. 25, No. 5, pp. 8-11, 2005

[3] G. Kanizsa, "Organization in Vision", *Publisher: Praeger*, New York, 1979

[4] Ruth Kimchi, Marlene Behrmann and Carl R. Olson, "Perceptual Organization in Vision: Behavioral and Neural Perspectives", *Publishers Lawrence Erlbaum Associates*, 2003, ISBN 0-8058-3872-4

[5] Merrian-webster online: http://www.m-w.com

[6] Andrew S. Winter, "Volume Graphics: Field-Based Modelling and Rendering", *PhD Thesis*, University of Wales, Swansea, Singleton Park, Swansea, SA2 8PP, Wales, UK, December, 2002

[7] Tamy Boubekeur, Patrick Reuter and Christophe Schlick, "Surfel Stripping", *Proc. Int. Conf. on Computer Graphics and Interactive Techniques in Australasia and Southest Asia, GRAPHITE '05*, Vol.3, pp. 177-186, 2005

[8] Min Chen, "Combining Point Clouds and Volume Objects in Volume Scene Graphs", *Proc. Volume Graphics, 2005*, pp. 127-135, 2005

[9] Julius Wiedemann, "Digital Beauties: 2D & 3D Computer Generated Digital Models, Virtual Idols and Characters", *Publisher Taschen*, 2001, ISBN:3-8228-1628-0

[10] Isaac V. Kerlow, "The Art of 3D Computer Animation and Effects", *Published by John Wiley & Sons, Inc.*, 2004, ISBN0-471-43036-6

[11] M. Froumentin, F. Labrosse and P. Willis, "A Vector-based Representation for Image Warping", *Computer Graphics Forum*, Vol. 19, No. 3, 2000, C385-C394, C428.

[12] V. Singh, D. Silver and N. Cornea, "Real-Time Volume Manipulation", *In Proc. Eurographics/IEEE TVCG Workshop on Volume Graphics 2003*, (Tokyo, Janpa, 2003), pp. 45-51,

[13] V. Singh and D. Silver, "Interactive Volume Manipulating with Selective Rendering for Improved Visualization", *In IEEE Symposium on Volume Visualisation and Graphics*, Oct., 2004, pp 95-102

[14] S. Islam, S. Dipankar, D. Silver and M. Chen, "Temporal and Spatial Splitting of Scalar Fields in Volume Graphics", *In Proc. IEEE VilVis2004*, (Austin, Texas, October 2004), pp 87-94

[15] C. Beeson and K. Bjorke, "Skin in the Dawn Demo", *Computer Graphics*, Vol.38, No.2, 2004, May, pp. 14-19

[16] Mark W. Jones, "The Visualisation of Regular Three Dimensional Data", *PhD Thesis*, University of Wales, Swansea, Singleton Park, Swansea, SA2 8PP, Wales, UK, December, 1995

[17] Richard A. Satherley, "Computation and Applications of Distance Fields in Volume Graphics", *PhD Thesis*, University of Wales, Swansea, Singleton Park, Swansea, SA2 8PP, Wales, UK, December, 2001

[18] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *In Computer Graphics (Proceedings of SIGGRAPH 87)*, Vol. 21, No. 4, pp. 163-169, July 1987

[19] T. Lewiner, H. Lopes, A. W. Vieira and G. Tavares, "Efficient Implementation of Marching Cubes Cases with Topological Guarantees", *Journal of Graphics Tools*, Vol. 8, No. 2, 2003, pp. 1-15.

[20] W. Heidrich, R. Seidel, H.-P. Wu, and T. Ertl, "Real-Time Exploration of Regular Volume Data by Adaptive Reconstruction of Iso-Surfaces", *The Visual Computer*, Vol. 15, No. 2, 1999, pp. 100-111.

[21] L. P. Kobbelt, M. Botsch, U. Schwanecke and H.-P. Seidel, "Feature Sensitive Surface Extraction from Volume Data", *In Proc. of ACM SIGGRAPH 2001*, pp. 57-66, 2001

[22] T. Ju, F. Losasso, S. Schaefer and J. Warren, "Data Contouring of Hermite Data", *In Proc. of Symposium on Volume Visualisation 2002*, pp. 339-346, 2002.

[23] S. F. F. Gibson, "Using Distance Maps for Accurate Surface Representation in Sampled Volumes", *In Proc. of Symposium on Volume Visualisation 1998*, pp. 23-30, 1998.

[24] E. Chernyaev, "Marching Cubes 33: Construction of Topologically Correct Isosurfaces", Technical Report CERN CN 95-17, CERN, 1995."

[25] Chien-Chang Ho, Fu-Che Wu, Bing-Yu Chen, Yung-Yu Chuang and Min Ouhyoung, "Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data", *In Eurographics 2005*, Dublin, Ireland, Vol. 24, No. 3, pp. 537-545.

[26] C. Zalhten and H. Jurgens, "Continuation Methods for Approximating Iso-Valued Complex Surfaces", In Computer Graphics Forum (Eurographics 91), Vol. 10, No. 3, pp. 5-19, 1991

[27] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade and D. Fulk, "The Digital Michelangelo Project: 3D Scanning of Large Status", *In Proc. SIGGRAPH 2000*, ACM Press, pp. 131-141, 2000.

[28] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler and L. McMillan, "Image-based Visual Hulls", *In Proc. SIGGRAPH 2000*, ACM Press, pp. 343-352, 2000.

[29] I. Guskov and Z. Wood, "Topological Noise Removal", *Graphics Interface 2001*, 2001, June, pp.19-26, Morgan Kaufmann

[30] N. Zhang, W. Hong and A. Kaufman, "Dual Contouring with Topology-Preserving Simplification Using Enhanced Cell Representation", *In Proc. of IEEE Visualisation 2004*, pp. 505-512, 2004.

[31] J. Davis, S.R. Marschner, M. Garr, M. Levoy, "Filling Holes in Complex Surface using Volumetric Diffusion", *Proc. 1st Int. Sym. On 3D Data Processing, Visualization, Transmission.* Padua, Italy, June 19-21, 2002

[32] Z. Wood, H. Hoppe, M. Desbrun, P. Schroder, An Out-of-core Algorithm for Isosurface Topology Simplification, ACM Trans. on Graphics, Vol.23, No.2, 2004, pp. 190-208

[33] Alla Sheffer, Bruno Levy, Maxim Mogilnitsky and Alexander Bogomyakov, "ABF++: Fast and Robust Angle based Flattening", *ACM Transactions on Graphics*, Vol. 24, No. 2, April, pp. 311-330, 2005.

[34] Suzanne M. Shontz and Stephen A. Vavasis, "A Mesh Warping Algorithm based on Weighted Laplacian Smoothing", *Proc. 12th. Int. Meshing Roundtable, IMR '03*, Vol. 12, pp. 1477-158, 2003.

[35] Olga Sorkine, "Laplacian Mesh Processing", *Proc. EUROGRAPHICS'05, STAR Report*, pp. 53-70, 2006

[36] M. Botsch and L. Kobbelt, "An Intuitive Framework for Real-time Freeform Modeling", *Proc. SIGGRAPH 2004*, pp. 630-634, 2004.

[37] Y. Limpman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossl and H.-P. Seidel, "Differential Coordinates for Interactive Mesh Editing", *Proc. of Shape Modeling International 2004*, IEEE Computer Society Press, pp. 181-190, 2004.

[38] Gil Zigelman, Ron Kimmel and Nahum Kiryati, "Texture Mapping Using Surface Flattening via Multidimensional Scaling", *IEEE Trans. on Visualization and Computer Graphics*, Vol.8, No.2, April-June, pp. 198-207, 2002.

[39] Peiyi Shen and Philip Willis, "Texture Mapping Volume Objects", *Proc. Vision, Video and Graphics, VVG'05*, Edinburgh, UK, Vol. 2, pp. 45-52, 2005

[40] Peiyi Shen and Philip Willis, "Texture for Animated Volume Characters", Proc. Int. Conf. on Computer Graphics and Interactive Techniques in Australasia and Southest Asia, GRAPHITE '05, Vol, 3, pp. 255-264, 2005

[41] S. Owada, F. Nielson, M. Okabe and T. Igarashi, "Volumetric Illustration: Designing 3D Models with Internal Textures", *Proceedings of SIGGRAPH 2004*, ACM Press / ACM SIGGRAPH, pp. 322-328, 2004

[42] David N. Rodgman, "Refraction in Volume Graphics", *PhD Thesis*, University of Wales, Swansea, Singleton Park, Swansea, SA2 8PP, Wales, UK, December, 2003

[43] C. Miller and M. W. Jones, "Texturing and Hypertexturing of Volumetric Objects", In E. Groller, I. Fujishiro, K. Mueller, T. Ertl (eds.), *Volume Graphics 2005*, Eurographics, pp. 117-125 [ISBN 3-905673-26-6] [ISSN 1727-8376]

[44] K. Perlin, "An Image Synthesizer", *Computer Graphics, (Proceedings of SIGGRAPH 85)*, Vol.19, No.3, 1985, July, pp. 287-296.

[45] D. R. Peachy, "Solid Texturing of Complex Surfaces", *Computer Graphics, (Proceedings of SIGGRAPH 85)*, Vol.19, No.3, 1985, July, pp. 279-286.

[46] http://www.nlm.nih.gov/research/visible/visible_human.html

[47] J. C. Hart, N. Carr, M. Kameya, S.A. Tibbits and J.J. Colemen, "Antialiased Parameterized Solid Texturing Simplified for Consumer-Level Hardware Implementation", In *SIGGRAPH/Eurographics Workshops on Graphics Hardware*, Aug., pp. 45-53, 1999.

[48] N. A. Carr, H. C. Hart and J. Maillot, "The Solid Map: Methods for Generating a 2-D Texture Map for Solid Texturing", *Manuscript, http://graphics.cs.uiuc.edu/ jch/papers/, eventually became "Meshed Atlases"*, Jan., 1999.

[49] N. A. Carr and J. C. Hart, "Meshed Atlases for Real-Time Procedure Solid Texturing", *ACM Transactions on Graphics*, Vol. 21, No. 2, 2002, Apr., pp. 106-131.

[50] A. S. Winter and M. Chen, "Image-Swept Volumes", *Computer Graphics Forum*, Vol. 21, No. 3, pp. 441-450, 2002.

[51] Li-Yi Wei, "Texture Synthesis by Fixed Neighbourhood Searching", *PhD Thesis*, Stanford University, 2001.

[52] D. J. Heeger and J. R. Bergen, "Pyramid-based Texture Analysis and Synthesis", *Proceedings of SIGGRAPH 1995*, pp. 229-238.

[53] G. Turk, "Texture Synthesis on Surfaces", *Proceedings of SIGGRAPH 2001*, pp. 347-354.

[54] E. Praun, "Lapped Textures", *Proceedings of SIGGRAPH 2000*, pp. 465-470.

[55] M. F. Cohen, J. Shade, S. Hiller and O. Deussen, "Wang Tiles for Image and Texture Generation", *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, pp. 287-294.

[56] L. Wang and K. Mueller, "Generating Sub-Resolution Detail in Images and Volumes Using Constrained Texture Synthesis", *Proc. IEEE Visualization 2004*, Oct., pp. 75-82, 2004.

[57] Aidong Lu and David S. Ebert, "Example-based Volume Illustrations", *IEEE Visualization 2005*, pp. 655-662, 2005

[58] Costa Sousa, M., Ebertd, D., Stredney, D., Svakhine, N., "Illustrative Visualization for Medical Training", *Proceedings of Computational Aesthetics 2005 - First Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pp. 201-209, 2005.

[59] Svakhine, N., Jang, Y., Ebert, D. S., Gaither, K., "Illustration and Photography-Inspired Visualization of Flows and Volumes", *IEEE Visualization 2005*, pp. 687-694, 2005.

[60] E. Bier and K. Sloan, "Two-Part Texture Mapping", In *IEEE Computer Graphics and Appliction*, Vol. 6, No. 9, pp. 40-53, 1986

[61] N. Greene, "Environment Mapping and Other Applications of World Projections", In *IEEE Computer Graphics and Application*, Vol. 6, No. 11, pp. 21-29, Nov., 1986.

[62] J. Kniss, G. Kindlmann and C. Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets", In *IEEE Visualisation Vis 2001*, pp. 255-262, 2001

[63] F.-Y Tzeng, E. B. Lum and K.-L, Ma, "A Novel Interface for Higher-Dimensional Classification of Volume Data", *In IEEE Visualisation 2003*, pp. 505-512, 2003.

[64] J. Allebach and P. W. Wong, "Edge Directed Interpolation", Vol. 3, In *Proc. IEEE Int. Conf. Image Processing 1996*, Sept., pp. 707-710, 1996.

[65] X. Li and M. Orchard, "New Edge Directed Interpolation", In *Proc. IEEE Int. Conf. Image Processing*, Vol. 2, Sept., pp. 311-314, 2000.

[66] D. Su and P. Willis, "Image Interpolation by Pixel Level Data-Dependent Triangulation", In *Computer Graphics Forum*, Vol. 23, No. 2, Jun., pp. 189-201, 2004.

[67] D. Su and P. Willis, "Demosaicing of Colour Images Using Pixel Level Data-Depedent Triangulation", In *Proc. EG UK Conference, Theory and Practice of Computer Science 2003*, Jun., pp. 16-23, 2003.

[68] Ivan Viola, Meister E. Gröller, Markus Hardwiger, Katja Bühler, Bernhard Preim and David Ebert, "Illustrative Visualization", *In Eurographics 2005 Tutorial 3: Illustrative Visualization*, Dublin, Ireland, pp. 7-15, 2005.

[69] Ivan Viola, Meister E. Gröller, Markus Hardwiger, Katja Bühler, Bernhard Preim and David Ebert, "Illustrative Visualization", *In Eurographics 2005 Tutorial 3: Illustrative Visualization*, Dublin, Ireland, pp. 49-55, 2005.

[70] A. Barr, "Ray Tracing Deformed Surfaces", *ACM Transactions on Graphics (Proceedings of SIGGRAPH 1986)*, pp. 287-296.

[71] M. Levoy, "Display of Surfaces from Volume Data", *ACM Transactions on Graphics (Proceedings of SIGGRAPH 1988)*, pp. 29-37.

[72] M. W. Jones, "Direct Surface Rendering of General and Genetically Bred Implicit Surfaces", *Proc. 17th Ann. Conf. of Eurographics (UK Chapter)*, pp. 37-46, 1999.

[73] M. Deering, S. Winer, B. Schediwy, C. Duffy and N. Hunt, "The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics", *In Proceedings of SIGGRAPH 1988*, pp. 21-30, 1988.

[74] A. Lastra, S. Molnar, M. Olano and Y. Wang, "Real-Time Programmable Shading", *In ACM Symposium on Interactive 3D Graphics*, pp. 59-66, 1995.

[75] T. Saito and T. Takahashi, "Comprehensible Rendering of 3D Shapes", *In Proceedings of SIGGRAPH 1990*, pp. 197-206, 1990.

[76] J. Hladuvka, A. Köng and E. Gröller, "Curvature-Based Transfer Functions for Direct Volume Rendering", *In Proceedings of Spring Conference on Computer Graphics 2000*, pp. 58-65, 2000.

[77] B. Lichtenbelt, R. Crane and S. Naqvi, "Introduction to Volume Rendering", Prentice Hall, 1998

[78] N. Max, "Optical Models for Direct Volume Rending", *IEEE Transactions on Visualisation and Computer Graphics*, Vol. 1, No. 2, pp. 99-108, 1995

[79] G. Kindlmann and J. W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering", *In Proceedings of IEEE Volume Visualisation 1998*, pp. 79-86, 1998

[80] S. Fang, T. Biddlecom and M. Tuceryan, "Image-Based Transferring Function Design for Data Exploration in Volume Visualisation", *In Proceedings of IEEE Volume Visualisation 1998*, pp. 319-326, 1998

[81] I. Fujishiro, T. Azuma and Y. Takeshima, "Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topological Analysis", *In Proceedings of IEEE Volume Visualisation 1999*, pp. 467-470, 1999

[82] J. Kniss, G. Kindlmann and C. Hansen, "Multidimensional Transfer Functions for Interactive Volume Rendering", *IEEE Transactions on Visualisation and Computer Graphics*, Vol. 8, No. 3, pp. 270-285, 2002

[83] G. Kindlmann, R. Whitaker, T. Tasdizen and T. Möller, "Curvature-based Transfer Functions for Direct Volume Rendering: Methods and Applications", *In Proceedings of IEEE Visualization 2003*, pp. 513-520, 2003.

[84] R. Crawfis and N. Max, "Texture Splats for 3D Scalar and Vector Field Visualisation", *In Proceedings of IEEE Visualisation 1993*, pp. 261-266, 1993

[85] N. Max, L.C. Leedom and R. Crawfis, "Flow Volumes for Interactive Vector Field Visualisation", *In VIS1993: Proceedings of the 4th conference on Visualisation 1993*, IEEE Computer Society, pp. 19-24, 1993

[86] S. Park, B. Budge, L. Linsen, B. Hamann and K. Joy, "Multi-Dimensional Transfer Functions for Interactive 3D Flow Visuzlization", *In Pacific Graphics*, pp. 513-520, 2004.

[87] David S. Ebert, "Interactive Volume Illustration for Medical and Surgical Training", *In Eurographics 2005 Tutorial 3: Illustrative Visualization*, Dublin, Ireland, pp. 60-68, 2005

[88] M. Chen J. Tucker, "Constructive Volume Geometry", *Computer Graphics Forum*, Vol. 19, No. 4, pp. 281-293, 2000.

[89] M. Chen and D. Silver and A. S. Winter and V. Singh and N. Cornea, "Spatial Transfer Functions – A Unified Approach to Specifying Deformation in Volume Modeling and Animation", *Proc. Volume Graphics 2003*, Eurographics/ACM Publications, Tokyo, Japan, PP. 35-44, 2003.

[90] Markus Hadwiger, "Illustrative Visualization of Isosurfaces and Volumes", *In Eurographics 2005 Tutorial 3: Illustrative Visualization*, Dublin, Ireland, pp. 26-47, 2005.

[91] H. Hauser, L. Mroz, G.-I. Bischi and E.Gröller "Two-level Volume Rendering", *IEEE Transactions on Visualisation and Computer Graphics 2001*, Vol. 7, No. 3, pp. 242-252, 2001.

[92] Ivan Viola, E. Gröller, "Smart Visibility in Visualisation", *In Eurographics 2005 Tutorial 3: Illustrative Visualization*, Dublin, Ireland, pp. 49-55, 2005

[93] P. Hasreiter, H. K. Cakmak and T. Ertl, "Intuitive and Interactive Manipulation of 3D Data Sets by Integrating Texture Mapping Based Volume Rendering into the Open Inventor Class Hierarchy", *In Bildverarbeitung in der Medizin: Algorithmen, Systeme, Anwendungen (1996)*, Spring Verlag, pp. 149-154, 1996.

[94] S. Pinker, "The Blank Slate: The Modern Denial of Human Nature", Viking Press, New York, 2003.

[95] YingCai Wu, "Effective, Intuitive, and Intelligent Volume Visualisation", *Postgraduate Qualification Exam (PQE) Report*, Dept. of Computer Science, Hongkong University of Science and Technology, Dec., 2005.

[96] M. Stone, "Color in Information Display: Principles, Perception, and Models", *In Course 20 in Proceedings of SIGGRAPH2004*, Aug., 2004.

[97] C. W. Victoria Interrante and Russell M. Taylor II, "Perceptually based Visualisation Design", *In Course 45 in Proceedings of SIGGRAPH2003*, July, 2003.

[98] C. Ware, "Information Visualisation: Perception for Design", *Publishers Morgan Kaufmann*, 2004.

[99] C. R. Johnson and A. R. Sanderson, "A Next Step: Visualising Errors and Uncertainty", *IEEE Comput. Graph. Appl.*, Vol. 23, No. 5, pp. 6-10, 2003.

[100] F. P. Vidal, F. Bello, K. W. Brodile, N. W. John, D. Gould, R. Philips and N. J. Avis, "Principles and Applications of Computer Graphics in Medicine", In *Computer Graphics Forum*, Vol. 25, No. 1, pp. 113-137, 2005.

[101] NIH/NSF Visualisation Research Challenges, Challenges Report, Jan., 2005, http://tab.computer.org/vgtc/vrc/index.html

[102] R. Kosara, C. G. Healey, V. Interrante, D. H. Laidlaw and C. Ware, "User Studies: Why, How, and When?", *IEEE Comput. Graph. Appl., 2003*, Vol. 23, No. 4, pp. 20-25, 2003.

[103] G. Grigoryan, "Point-based Probalistics Surfaces to Show Surface Uncertainty", *IEEE Transactions on Visualisation and Computer Graphics*, Vol. 10, No. 5, pp. 564-573, 2004.

[104] Chris Johnson, Robert Moorhead, Tamara Munzner, Hanspeter Pfister, Penny Rheingans, and Terry S. Yoo, "NIH/NSF Visualization Research Challenges Report - January 2006" http://tab.computer.org/vgtc/vrc/index.html.

[105] O. Polonsky, G. Patanè, S. Biasotti, G. Gotsman, and M. Spagnuolo, "What's in an Image", *The Visual Computer*, Vol. 21, No.8-10, pp. 840-847, 2005.

[106] Y. T. Shigeo Takahashi, Issei Fujishiro and T. Nishita, "A Feature-Driven Approach to Locating Optimal Viewpoints for Volume Visualisation", *In Proceedings of IEEE Visualiation 2005 (VIS'05)*, pp. 495-502, 2005.

[107] Han-Wei Shen and Udeepta D. Bordoloi, "View Selection for Volume Rendering", *In Proceedings of IEEE Visualiation 2005 (VIS'05)*, pp. 62-69, 2005.

[108] F.-Y Tzeng, "An Intelligent System Approach to Higher-Dimensional Classification of Volume Data", *In IEEE Transactions on Visualiation and Computer Graphics 2005*, Vol. 11, No. 3, pp. 273-284, 2005.

[109] F.-Y Tzeng and K.-L, Ma, "A Cluster-Space Visual Interface for Arbitrary Dimensional Classification of Volume Data", *In IEEE TVCG Symposium on Visualisation*, pp. 17-24, 338, 2004

[110] M. E. Froumentin and P. J. Willis, "An Efficientt 2.5D Rendering and Compositing System", In *Computer Graphics Forum - Conjference Issue*, Vol. 18, No. 3, pp. C385–C394, C428, Sept. 1999.

[111] M. Qi and P. Willis, "Quasi-3D Cel-based Animation", In *Proc. of Vision, Video and Graphics 2003*, Bath, UK, Jul., pp. 111-116, 2003.

[112] J. T. Kajiya and B. P. Von Hersen, "Ray Tracing Volume Densities", In *Computer Graphics*, Vol. 18, No. 3, Jul., pp. 165-174,, 1984.

[113] Tomihisa Welsh, Michael Ashikhmin and Klaus Mueller, "Transferring Colour to Greyscale Image", *SIGGRAPH2002*, 2002.

[114] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley, "Color Transfer between Images", *IEEE Computer Graphics and Applications*, Vol. 21, No. 5 (2001) (special issue on Applied Perception), pp. 34-41.

[115] Aidong Lu, David S. Ebert, "Example-based Volume Illustrations", *IEEE Visualisation 2005*, pp. 655–662, October, 2005.

[116] Amy A. Gooch, Sven C. Olsen, Jack Tumblin, Bruce Gooch, "Color2Gray: Salience-Preserving Color Removal", *SIGGRAPH2005*, Volume 24, Issue 3, pp. 634-639, 2005.

[117] E. Ferley and M.-P. Cani and J.-D. Gascuel, "Practical Volumetric Sculpting", In *The Visual Computer*, Vol. 16, pp. 469-480, 2000.

[118] T. Tasdizen and R. Whitaker and P. Burchard and S. Osher, "Geometric Surface Smoothing via Anisotropic Diffusion of Normals", In *IEEE Vis2002*, Oct., pp. 125-132, 2002.

[119] Bochev, P., Liao, G. and dela Pena, G., "Analysis and computation of adaptive moving grids by deformation", *Numer. Methods Partial Diff. Equat.*, Vol.12, No.4, pp. 489-506, 1996.

[120] Berzins, M., Durbeck, LJK., Jimack, P. K., Walkley, M., "Mesh quality and moving and meshes for 2D and 3D unstructured mesh solvers", Weatherill, N P and Deconink, H (editors), *Von Karman Institute for Fluid Mechanics Lectures notes for 31st Lecture Series on Computational Fluid Mechanics Von Karman Institute for Fluid Mechanics*, 2000.

[121] Joseph B. Kruskal and Myron Wish, "Multidimensional Scaling", *SAGE Publications*, 1978, ISBN 0803909403.

[122] Ruth Grossmann, Nahum Kiryati and Ron Kimmel, "Computational Surface Flattening: A Voxel-Based Approach", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.24, No.4, April, pp. 433-441, 2002.

[123] Nahum Kiryati and Olaf Kübler, "Chain Code Probabilities and Optimal Length Estimators for Digital Three Dimensional Curves", *Pattern Recognition*, Vol. 28, pp. 361-372, 1995.

[124] N. Kiryati and G. Székely, "Estimating Shortest Paths and Minimal Distances on Digitized Three-Dimensional Surfaces", *Pattern Recognition*, Vol. 26, pp. 1623-1637, 1993.

[125] Andreas Buja and Deborah F. Swayne, "Visualisation Methodology for Multidimensional Scaling",*J. of Classification*, Vol.19, pp. 7-43, 2002

[126] Andreas Buja, Deborah F. Swayne, Michael L. Littman, Nathaniel Dean and Heike Hofmann, "Interactive Data Visualisation with Multidimensional Scaling", http://www-stat.wharton.upenn.edu/ buja/PAPERS/paper-mds-jcgs.pdf.

[127] Lisha Chen and Andreas Buja, "Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Layout and Proximity Analysis", PhD Thesis, University of Pennsylvania, 2006.

[128] J. A. Bondy and U. S. R. Murty, "Graph Theory and Application", *The Gresham Press* , 1976, ISBN 0333177916.

[129] Yunjin Lee, Hyoung Seok Kim and Seungyong Lee, "Mesh Parameterization with a Virtual Boundary", *Computers & Graphics*, Vol.26, pp. 677-686, 2002.

[130] Alla Sheffer, Emil Praun and Kenneth Rose, "Mesh Parameterization Methods and their Applications", *Foundations and Trends in Computer Graphics and Vision, Published by Now Publishers*, ISBN 978-1-933019-43-7, Dec., 2006.

[131] Floater, M.S., "Parameterization and Smooth Approximation of Surface Triangulations", *Computer Aided Geometric Design*, Vol.14, pp. 231-250, 1997.

[132] Craig Gotsman, Xiangfeng Gu and Alla Sheffer, "Fundamentals of Speherical Parameterisation for 3D Meshes", *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, Vol.22, No.3, pp. 358-363, 2003.

[133] Willie Brink, "Spherical Parameterisation Methods for 3D Surfaces", Master Thesis, University of Stellenbosch, Dec., 2005.

[134] James Foley, Andries van Dam, Steven Feiner and John Hughes, "Computer Graphics", 2nd Edition, 1990, Addison-Wesley, ISBN 0-201-12110-7.

[135] D. Lischinski and A. Rappoport, "Image-Based Rendering for Non-Diffuse Synthetic Scenes.", In G. Drettakis and N. Max, editors, *Eurographics Workshop on Rendering 1998*, Eurographics, Springer Wien, 1998, New Orleans, Louisiana, 04-09 August, pp. 301-314, 1998.

[136] Christian Tietjen, Tobias Isenberg and Bernhard Preim, "Combining Silhouettes, Surfaces, and Volume Rendering for Surgery Education and Planning", in *Proceeding of EUROGRAPHICS IEEE VGTC Symposium on Visualisation 2005*, K.W. Brodlie, D.J.Duke and K.I. Joy (Editors), ISBN:3905673193, pp. 303-310, 2005.

[137] Jerome Maillot, Hussein Yahia, and Anne Verroust, "Interactive Texture Mapping", In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, pp. 27-34, 1993.

[138] Bruno Levy, Sylvain Petitjean, Nicolas Ray and Jerome Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generaion", *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, Jul., Vol.21, No.3, pp. 362-371, 2002.

[139] A. Sheffer and E. de Sturler, "Param. of Faceted Surfaces for Meshing Using Angle-based Flattening", *Engineering with Computers*, Vol.17, No.3, pp. 326-337, 2001.

[140] http://www.ljmu.ac.uk/GERI/MEGURATH.htm

[141] C. Schmid, R. Mohr and C. Bauckhage, "Evaluation of Interest Point Detectors", *International Journal of Computer Vision*, Jun., Vol.37, No.2, pp. 151-172, 2000.

[142] P.H.S. Torr, "A Structure and Motion Toolkit in Matlab: Interactive Adventures in S and M", http://research.microsoft.com/ philtorr/

[143] C. Harris and M.J. Stephens, "A Combined Corner and Edge Detector", In *Proceedings of The Fourth Alvey Vision Conference, Manchester*, pp. 147-152, 1988.

[144] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Application to Image analysis and automated cartography", *Commun. Assoc. Comp. March.*, Vol.24, pp. 381-395, 1981.

[145] Xiaoguang Lu, Anil K. Jain and Dirk Colbry, "Matching 2.5D Face Scans to 3D Models", IEEE Transaction on Pattern Analysis and Machine Intellegence, Vol. 28, No. 1, pp. 31-43, 2006.

[146] C. Huntzinger, P. Munro, S. Johnson, M. Miettinen, C. Zankowski, G. Ahlstrom, R. Glettig, R. Filliberti, W. Kaissl, M. Kamber, M. Amstutz, L. Bouchet, D. Klebanov, H. Mostafavi, and R. Stark "Dynamic Targeting Image-Guided Radiotherapy", in *Medical Dosimetry*, Vol. 31, No. 2, pp. 113-125, 2006.

[147] G. Scott and H. C. Longuet-Higgins, "An Algorithm for Associating the Features of Two Images", In *Proceeding of Royal Society of London*, B(244):21–26, 1991.

[148] Maurizio Pilu, "A Direct Method for Stereo Correspondence based on Singular Value Decomposition", IEEE Computer Vision and Pattern Recognintion Conference, CVPR'97, pp. 261-266, 1997.